

Phạm Huy Hoàng

Tôi Đi Code Dạo

NHẬP MÔN LẬP TRÌNH KHÔNG CODE

Lời tựa

Từ câu hỏi của nhiều bạn trẻ

Hiện nay, ngành lập trình đang là một ngành hot, nhận được sự chú ý của nhiều bạn trẻ.

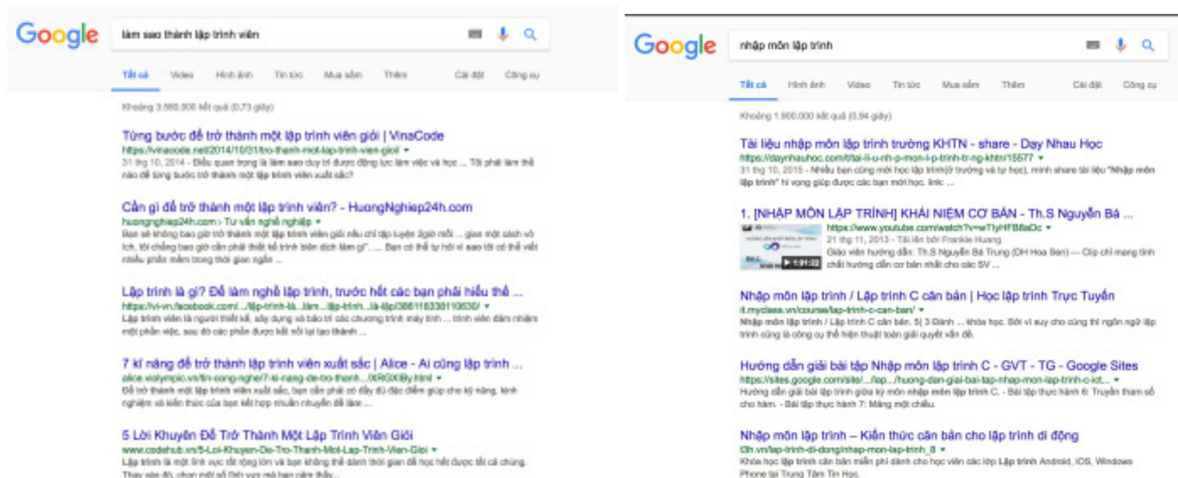
Từ lúc viết blog đến này mình vẫn thường nhận được câu hỏi "Làm sao để trở thành một lập trình viên", hoặc "Em đi học rồi nhưng vẫn còn đang mù mờ không biết ngành này thế nào, nên học những gì?".

Những câu hỏi này thường đến từ các bạn học sinh sắp thi Đại Học, sinh viên năm nhất năm hai, hoặc những bạn đã tốt nghiệp, đã đi làm nhưng có hứng thú muốn tìm hiểu về ngành IT.

Số lượng câu hỏi mình nhận được không hề ít, cho thấy có rất nhiều bạn quan tâm đến ngành IT và muốn theo đuổi nó.

Đến ebook Nhập Môn Lập Trình Không Code

Thế nhưng, khi mình thử tìm hiểu về ngành lập trình dưới góc độ của một newbie, mình mới thấy nó… không hề dễ dàng một tí nào. Đa phần các tài liệu đều nặng về code, bập một phát là đưa ra lý thuyết, bắt tay ngay vào code làm nhiều bạn hoảng hồn.



Trong quá trình làm việc, lập trình viên **dành phần lớn thời gian cho việc code**, nhưng công việc của lập trình viên **không phải chỉ có code**!

Theo mình, trước khi dạy code, cần phải có **định hướng cho các bạn** về ngành lập trình như: công việc của lập trình viên, những tố chất cần có, lương bổng và cơ hội thế nào…

Do vậy, mình chọn một hướng tiếp cận riêng, nhập môn lập trình mà **không đụng đến một dòng code nào**. Điều này sẽ giúp các bạn đọc đỡ ngộp, đỡ sợ hơn khi tìm hiểu ngành này.

Quyển sách này có gì hay?

Nếu không nói về code, vậy quyển sách này nói về cái gì? Bạn hãy coi nó là một cuốn sách **định hướng nghề nghiệp.** Đối tượng mà ebook này hướng đến là **các em lớp 12, những bạn năm nhất đại học hoặc những bạn đang học ngành nghề khác**, muốn tìm hiểu về ngành lập trình.

Tuy nhiên, dù bạn đã chọn theo đuổi ngành CNTT, bạn vẫn có thể đọc series để biết mình còn thiếu những kiến thức gì, cần học thêm những gì, đi làm khác đi học ra sao… Chưa kể, sau khi đọc xong, bạn cũng sẽ biết cách hướng dẫn bạn bè muốn học lập trình hoặc định hướng cho đàn em chẳng hạn.



Đây là những thắc mắc mà sách sẽ giải đáp cho bạn sau:

- Làm lập trình viên là làm gì? Công việc thường ngày của họ là gì?
- Triển vọng nghề nghiệp và lương bổng của ngành lập trình.
- Học lập trình cần những tư chất gì? Liệu bạn có phù hợp với ngành lập trình không?
- Học lập trình có thể làm được gì: Phần mềm, app di động, web, game…
- Hai con đường làm lập trình viên: Đại Học và Học Đại (Tự học, thấy gì học nấy)
- Làm sao học ngôn ngữ lập trình đầu tiên?
- Học "xong" ngôn ngữ lập trình đầu tiên thì làm gì?
- Kĩ năng mềm cứng mà lập trình viên phải biết: làm việc nhóm, tiếng Anh, tự học, ….

P/S: Đây là ebook miễn phí, các bạn cứ thoải mái chia sẻ cho bạn bè, người thân, nhớ dẫn nguồn toidicodedao.com là được nhé. Để ủng hộ tác giả, nhớ ghé thăm và like fanpage tại: https://www.facebook.com/toidicodedao nhe.

Các bạn hãy click **Đăng kí nhận email** để theo dõi blog và nhận những **ebook miễn phí**, những **bài viết cực kì hay ho hàng tuần** về kĩ năng mềm và cứng, kinh nghiệm trong ngành lập trình nhé!

Đăng kí nhận email

Mục lục

Công việc thường ngày của một lập trình viên

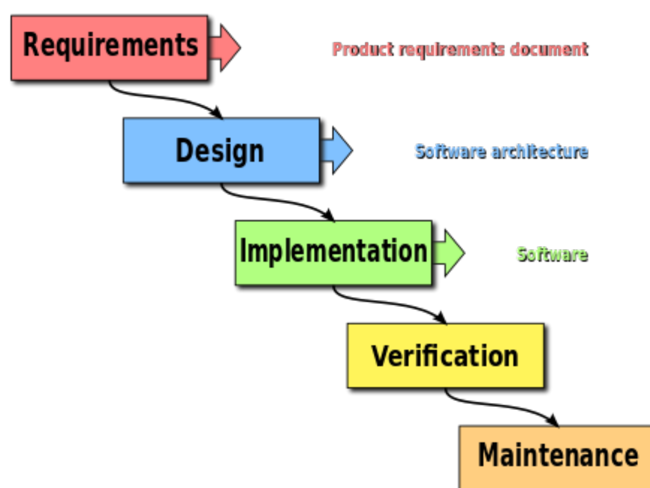
Chương này sẽ trả lời câu hỏi "Làm lập trình viên là làm gì?" và giới thiệu những công việc hằng ngày của mỗi lập trình viên.

Làm lập trình viên là làm gì?

Nói một cách đơn giản, công việc của lập trình viên là "lập trình", tức là **viết code để tạo ra phần mềm**. Phần mềm ở đây có thể là ứng dụng di động (Google Map, Camera 360), game (Flappy Bird, Angry Bird), web (Facebook, Instagram), ứng dụng Window (ứng dụng quản lý, bán hàng…) hoặc các hệ thống nội bộ cho các công ty.

Qui trình cơ bản để tạo ra một phần mềm thường bao gồm: business analysis (BA) phụ trách **phân tích nghiệp vụ và lấy yêu cầu** của khách hàng; designer để **thiết kế các màn hình và luồng chạy**; lập trình viên (developer) sẽ dựa vào đó để **viết code tạo nên chương trình**; sau đó tester sẽ **chạy thử để tìm lỗi** và… quăng cho developer sửa.

Với các web, ứng dụng nho nhỏ, lập trình viên sẽ tự mình làm hết các công đoạn trên, từ lấy yêu cầu khách hàng cho tới design và viết code, sau đó test thử sản phẩm.



Quy trình phát triển một phần mềm

Công việc hàng ngày của mỗi lập trình viên

Tới đây chắc bạn cũng hiểu sơ về công việc của mỗi lập trình viên. Tuy nhiên, nhiều bạn sinh viên vẫn không rõ khi đi làm mình sẽ phải làm những việc gì.

Làm một lập trình viên, công việc thường ngày của bạn đa phần là xoay quanh chiếc máy tính:

- **Code**: Phần lớn thời gian đi làm của bạn là dùng để code. Ở vị trí junior hoặc developer, bạn sẽ code những hàm hoặc chức năng nhỏ. Ở các vị trí cao hơn bạn sẽ **nhận nhiều trách nhiệm hơn**, code các module phức tạp hơn.

- **Test**: Thông thường, sau khi code xong một chức năng nào đó, ta sẽ đưa cho tester kiểm thử để tìm lỗi. Tuy vậy, trước khi đưa cho tester, ta cũng phải chạy thử và viết unit test cẩn thận để chắc chắn chương trình chạy đúng, module đã hoàn thành.
- **Fix bug**: Bug là những lỗi ta gặp khi code, làm chương trình chạy sai. Code thì lúc nào cũng có bug, không ít thì nhiều. Khi phát hiện bug, ta phải vọc và sửa code để chương trình chạy đúng.
- **Học cái mới**: Đôi khi ta phải tham gia một dự án sử dụng công nghệ mới hoặc công nghệ… quá cũ mà ta không biết. Lúc này ta phải tự học công nghệ đó (thông qua ebook, khoá học online) để có thể làm việc được.



Lập trình viên không phải chỉ biết code

Tuy nhiên, như mình đã nói, lập trình viên **không phải lúc nào cũng code**. Ngoài code ra, ta còn phải làm khá nhiều việc không dính dáng tới máy tính như:

- **Suy nghĩ**: Trước khi đặt bút viết code, nhầm, gõ code, ta cần phải ngồi phác thảo và suy nghĩ hướng giải quyết. Việc suy nghĩ cẩn thận trước khi code rất quan trọng, nó giúp bạn có cái nhìn tổng quát vấn đề, không bỏ quên các trường hợp thừa.
- **Phân tích/Thiết kế**: Với các module phức tạp, trước khi code bạn phải làm việc với đồng đội cùng team để phân tích rõ ràng, thiết kế các module trước khi code. Việc này khá là vui, hồi bên UK cứ mỗi lần cần thảo luận là mình và teammate lại kiếm cái bảng, vừa viết viết vẽ vẽ phân tích vừa chém gió.
- **Họp hành và báo cáo**: Theo qui trình Scrum, mỗi ngày bạn sẽ mất khoảng 10 phút tham gia họp Scrum (Daily Meeting) để báo cáo về những việc mình đã/sẽ làm. Ngoài ra, bạn còn phải tham dự đủ thứ cuộc họp liên quan đến thiết kế hệ thống, báo cáo tình hình, họp demo cho khách hàng.
- **Giao tiếp với khách hàng/stackholder**: Theo lý thuyết thì BA sẽ giao tiếp với khách hàng và lấy requirement, developer chỉ việc code. Tuy nhiên, trên thực tế, ở các công ty hoặc team nhỏ, đôi khi **chính developer phải nói chuyện với khách hàng** để làm rõ yêu cầu, demo sản phẩm. Thời còn làm FPT, mình vẫn phải lên forum để hỏi khách hàng và bên designer về những phần chưa rõ.



Kết luận

Ở phần này, chúng ta đã tìm hiểu những việc mà hầu như developer nào cũng làm hằng ngày trong khoảng thời gian đầu đi làm.

Khi bạn lên các vị trí cao hơn, thời gian code sẽ ít đi, thay vào đó bạn sẽ bỏ nhiều thời gian hơn để phân tích thiết kế, phỏng vấn developer nếu bạn theo hướng technical; hoặc bỏ nhiều thời gian hơn để quản lý, giao tiếp với khách hàng nếu bạn đi theo hướng management.

Ở chương sau, chúng ta sẽ tìm hiểu về **những tư chất cần có để theo ngành lập trình**? Làm sao biết liệu bạn có phù hợp với ngành lập trình hay không?

Những tố chất cần có để trở thành lập trình viên

Nhiều bạn có hỏi mình là "Muốn làm lập trình viên **cần có những tư chất gì**? Làm sao để biết mình **có phù hợp với ngành** này hay không?" Chương này sẽ giải đáp những thắc mắc nói trên, đồng thời dẫn ra những sai lầm mà nhiều người thường nghĩ về lập trình viên nhé.

Hiểu lầm thường gặp về lập trình viên

Do hậu quả của báo chí và phim ảnh (Tấm gương Bill Gates, Mark Zuckerberg hoặc phim Mr. Robot, The Social Network, …), một số bạn học sinh sinh viên thường có những lầm tưởng sau về lập trình viên:

- **Muốn làm lập trình viên thì phải cực kỳ thông minh cỡ… thiên tài**: Sai! Bạn không cần phải giỏi như Bill Gates hay Mark Zuckerberg để có thể làm lập trình viên, chỉ cần có một số tố chất là được (xem phần dưới).
- **Muốn làm lập trình viên phải giỏi Toán**: Không hẳn là đúng! Giỏi toán sẽ giúp bạn **suy nghĩ logic** tốt hơn, code tốt hơn. Tuy vậy, công việc lập trình thường rất **ít khi sử dụng các kiến thức toán cấp cao** (tích phân, đạo hàm, ma trận…), chỉ cần cộng trừ nhân chia và logic. Tuy viên, cũng có một số lĩnh vực chuyên biệt cần sử dụng nhiều Toán như developer game, data mining, machine learning, ứng dụng giả lập v…v
- **Lập trình viên thường ù lì, ít nói, thích làm việc một mình**: Sai! Lập trình là một công việc tập thể, đòi hỏi giao tiếp nhiều nên không có chuyện lập trình viên chỉ cắm đầu vào máy code một mình là xong việc.



Tố chất cần có để theo đuổi ngành lập trình

Không cần phải là thiên tài, cũng **không cần phải giỏi toán**, vậy bạn cần những gì để có thể thành một lập trình viên? Bạn cần những tố chất sau đây:

- **Khả năng suy nghĩ logic, giải quyết vấn đề**: Công việc lập trình đa phần giống như giải đố, và người lập trình viên viết code hoặc sử dụng thư viện/framework có sẵn để

giải quyết vấn đề đó. Các bạn có thể thử một bài test khả năng logic ở đây: Test logic (Đề thi tuyển vào ĐH FPT cũng bao gồm 105 câu hỏi logic dạng này).

- **Tính kiên nhẫn**: Việc học lập trình đòi hỏi tính kiên nhẫn rất cao. Việc code cũng thế, đôi khi bạn sẽ mất cả buổi trời để tìm một con bug hoặc sửa một lỗi nhỏ. Nếu không đủ kiên nhẫn bạn sẽ rất dễ bỏ cuộc.
- **Khả năng hoà đồng, kĩ năng giao tiếp**: Lập trình là một công việc tập thể, bạn sẽ phải làm việc chung với các thành viên khác (từ trưởng nhóm, developer cho tới tester). Do đó kĩ năng giao tiếp, làm việc nhóm là không thể thiếu.
- **Tinh thần tự giác**: Khi đi làm, thông thường trưởng nhóm sẽ không cầm tay chỉ việc mà chỉ giao việc, bạn sẽ phải tự giác sắp xếp thời gian, tìm hiểu công nghệ để thực hiện. Công nghệ mới liên tục thay đổi nên phải có **tinh thần tự giác và đam mê** thì bạn mới có thể cập nhật kiến thức cho bản thân, giữ cho mình không lạc hậu.
- **Tính tỉ mẩn, để ý tiểu tiết**: Để viết ra chương trình tốt, ít lỗi, developer phải để ý đến những tiểu tiết khi code, không bỏ dỡ những trường hợp ít gặp. Việc để ý tiểu tiết sẽ giúp bạn viết code ít lỗi hơn, thiết kế tổ chức code tốt hơn.
- **Lười biếng**: Tuy khó tin nhưng đây là một phẩm chất mà developer nên có. Thay vì bỏ thời gian công sức ra cày cuốc OT, viết code nhiều, lập trình viên cần phải hơi "làm biếng" để tìm ra hướng giải quyết **nhanh chóng và ít tốn công sức** hơn.



Tất nhiên, để trở thành một lập trình viên, bạn không cần toàn bộ những tố chất phía trên mà **chỉ cần phần lớn**. Có những coder code và thiết kế giỏi nhưng rất ngại giao tiếp; hoặc có những bạn dev giải quyết vấn đề rất nhanh nhưng lại hơi ẩu, thiếu tỉ mẩn nên code hay mắc lỗi.

Nếu bạn có một vài đức tính trong danh sách này, cộng với đam mê với ngành phần mềm thì **cứ dấn thân thôi, đừng ngại ngần gì nhé!**

Những thái độ không phù hợp với ngành lập trình

Nếu có một số thái độ hoặc cách nghĩ dưới đây, bạn không nên theo đuổi ngành lập trình mà **hãy chọn ngành khác** phù hợp với bản thân mình hơn:

- Thiếu tự giác, muốn được hướng dẫn công việc cụ thể
- Thiếu kiên nhẫn, không thích tự tìm tòi cái mới

- Thích làm việc cá nhân, ghét giao tiếp và làm việc nhóm
- **Muốn ngày làm 8 tiếng, giờ giấc ổn định**: Trong ngành lập trình, việc OT (overtime tức làm thêm giờ) khá phổ biến. Những khi dự án vào giao đoạn khẩn cấp, cả đội ngũ phải **làm thêm tới 8-9h tối hoặc T7-CN** nên giờ giấc cũng khá thất thường.
- **Muốn làm giàu nhanh**: Lương của lập trình viên cao hơn mặt bằng chung một chút nhưng cũng chỉ đủ sống. Các trường hợp giàu có bất ngờ như Nguyễn Hà Đông hoặc giàu có nhờ startup **cũng có nhưng rất hiếm**. Đa phần lập trình viên vẫn phải đi làm 8 tiếng một ngày, cuối tháng nhận lương như bao ngành nghề bình thường khác.



Kết

Bài viết này chia sẻ một số sai lầm thường gặp khi nói về lập trình viên, những tố chất cần có và không nên có nếu muốn theo đuổi ngành lập trình. Nếu muốn bổ sung điều gì các bạn cứ thoải mái comment nhé!

Ở bài viết sau, mình sẽ nói chuyện "Học lập trình có thể làm được những gì?", đồng thời chia sẻ một số điều về triển vọng nghề nghiệp và lương bổng trong ngành lập trình. Các bạn nhớ follow fanpage và đón xem nha.

Triển vọng nghề nghiệp của ngành lập trình

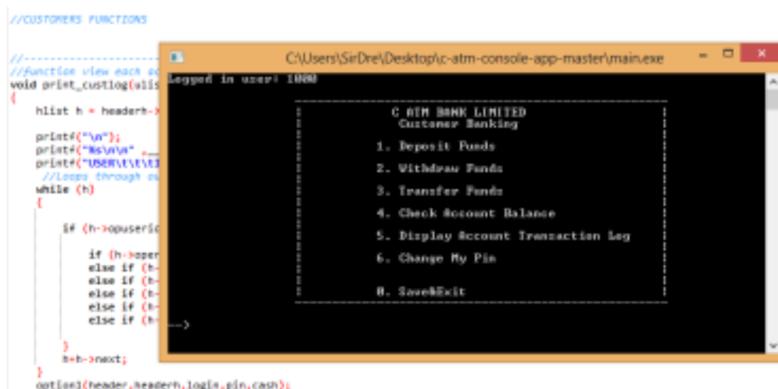
Ở chương trước, mình đã chia sẻ về những tố chất mà mỗi lập trình viên nên có. Bên cạnh đó, khi tham dự một số hội thảo hướng nghiệp, mình vẫn thường nghe các bạn hỏi những câu hỏi như:

- Học lập trình thì làm ra được gì?
- Làm ngành này ra trường có dễ xin việc không?
- Lương có cao không?
- Em nghe nói tuổi nghề chỉ khoảng 30-40, có thật không?

Chương này sẽ giải đáp những thắc mắc nói trên xen lẫn với một số lời khuyên dựa theo kinh nghiệm bản thân mình nhé!

Học lập trình thì làm ra được cái gì?

Trong các trường đại học, người ta thường dạy C, C++ trước tiên cho sinh viên để giúp họ tiếp cận với lập trình. Ở giai đoạn đầu, các bạn chủ yếu viết chương trình trên màn hình console nên họ tưởng rằng lập trình chỉ có thể viết mấy cái nho nhỏ, lặt vặt nên… mau chán.



Khi mới học lập trình, các bạn developer thường code các ứng dụng console như thế này

Thật ra, sau khi học lập trình, bạn có thể làm được những thứ từ nhỏ xíu đến cực to như sau:

- **Ứng dụng Windows, hệ thống phần mềm doanh nghiệp**: Từ các hệ thống nhỏ như quản lý khách sạn, tính tiền bán hàng cho tới các hệ thống lớn như quản lý kho hàng, core banking, hệ thống bán vé máy bay.
- **Ứng dụng di động**: Ứng dụng trên các hệ điều hành Android, iOS như Facebook, Instagram.
- **Web**: Từ các website tin tức, bán hàng như kenh14.vn, amazon.com tiki.vn cho đến các ứng dụng web (web app) phức tạp như Google, Dropbox.
- **Embedded software**: Thiết kế vi mạch và viết code lập trình cho các mạnh này.
- **Khác**: Một số mảng khác cũng khá hay như lập trình game, lập trình hệ thống trí tuệ nhân tạo (AI), khai thác dữ liệu (data mining)…

Với các bạn ngành khác, khi có ý tưởng, họ phải đi tìm người hỗ trợ để tạo ra ứng dụng. Ngược lại, lập trình viên **rất dễ khởi nghiệp** vì họ có thể sử dụng chính kĩ năng của mình để hiện thực ý tưởng của mình.

Nếu có kĩ năng lập trình vững, bạn hoàn toàn có thể tự mình làm Flappy Bird như Nguyễn Hà Đông, hoặc tự làm một sản phẩm dựa theo ý tưởng bản thân. Bạn cũng có thể được **mời làm tech co-founder** cho startup nào đó, sướng chưa!



Không có anh tech co-founder bên trái thì Steve Jobs tuổi gì mà tạo ra Apple nổi

Nếu không muốn khởi nghiệp, bạn có thể kiếm việc làm trong các công ty và dần dần leo lên vị trí cao hơn. Đọc đoạn dưới để tìm hiểu về nhu cầu thị trường nhé!

Ra trường có dễ xin việc không?

Câu trả lời là **CÓ**, ra trường bạn không cần quan hệ hay chi tiền lót tay gì cả, chỉ cần có CV ổn và kĩ năng tốt là kiếm được việc.

Rất nhiều bài báo nói về sự thiếu hụt nhân lực và triển vọng của ngành CNTT. Chỉ cần chịu khó Google 1 tí, các bạn sẽ thấy các công ty rất "khát" nhân lực và luôn trong tình trạng tuyển dụng.

Nhân lực ngành công nghệ thông tin: sẽ còn "khát" dài dài... - Tuổi Trẻ ...
tuoitre.vn/tin/nhip-song-tre/...luc...nghe...tin.../1149487.html - Translate this page
Nhân lực ngành công nghệ thông tin: sẽ còn "khát" dài dài... 05/08/2016 12:22 GMT+7. TTO - Theo dữ liệu được Công ty tuyển dụng trực tuyến VietnamWorks ...

Thiếu hụt nhân lực công nghệ thông tin ở mức báo động đỏ - VnExpress
vnexpress.net › Giáo dục ▾ Translate this page
Jun 29, 2015 - Thiếu hụt nhân lực công nghệ thông tin ở mức báo động đỏ ... Nhưng buồn thay ngành CNTT Việt Nam lại thiếu nguồn lực để triển khai.

Nhân lực công nghệ thông tin Việt Nam – Wikipedia tiếng Việt
https://vi.wikipedia.org/.../Nhân_lực_công_nghệ_thông_tin_Việt_... ▾ Translate this page
Jump to Một số trường đại học, cao đẳng đào tạo nhân lực ngành công nghệ ... - Một số trường tiêu biểu về đào tạo ngành Công nghệ Thông tin bao gồm:.

Nhân lực ngành công nghệ thông tin: Khoảng trống chưa thể lấp đầy ...
hbu.edu.vn/.../nhan-luc-nganh-cong-nghe-thong-tin-khoang-trong... ▾ Translate this page
Ngày đăng: 09:16:59-08/08/2016. Theo dữ liệu thống kê trong năm 2015 của Công ty tuyển dụng trực tuyến VietnamWorks, số lượng việc làm nhóm ngành ...

Thiếu hụt nhân lực công nghệ thông tin ở mức báo động đỏ | Trang ...
https://tuyensinh.uit.edu.vn/thieu-hut-nhan-luc-cong-nghe-thong-ti... ▾ Translate this page
Jun 30, 2015 - Thiếu hụt nhân lực công nghệ thông tin ở mức báo động đỏ ... cấp 32.000 sinh viên tốt nghiệp CNTT và các ngành có liên quan đến CNTT.

Tình trạng "khát" nhân lực trong ngành CNTT là có thật

Đây hoàn toàn là sự thật chứ không phải báo chí "chém" ra. Tuy vậy **đời không phải màu hồng** nên các bạn đừng quá chủ quan! Các công ty **rất cần người** nhưng không phải ai họ cũng tuyển, mà chỉ tuyển những nhân sự có chất lượng (Có khả năng technical vững, thái độ làm việc tốt, chịu khó học hỏi, ...).

Có một nghịch lý là: dù thị trường đang khát nhân lực nhưng nhiều sinh viên ra trường vẫn **không kiếm được việc làm** vì không đủ kĩ năng (Hậu quả của việc khi đi học không chịu tự code đi nhờ giải bài tập hộ). Đơn cử như thằng bạn mình kể từng phỏng vấn hơn 20 ứng viên mà cuối cùng chỉ chọn được 2 bạn vào làm việc.

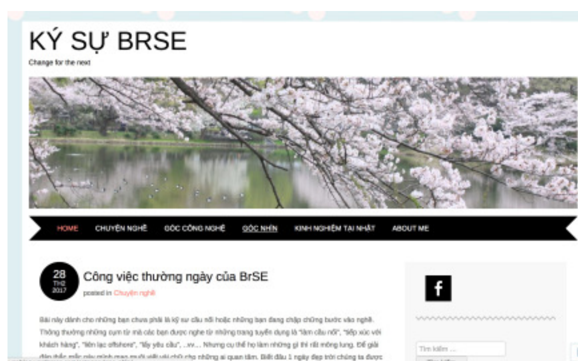
Do đó, bạn phải chuẩn bị học tập, rèn luyện kĩ năng lập trình, tiếng Anh và kĩ năng mềm ngay từ lúc còn ngồi trên ghế nhà trường. Đừng nghĩ rằng cứ học tàn tàn ra trường là sẽ có công ty hốt ngay nhé! (Các bạn tốt nghiệp từ đại học danh tiếng như BK, KHTN thường có nhiều cơ hội hơn).

Lương bổng và cơ hội phát triển bản thân

Mình từng có một bài viết về hướng phát triển và mức lương trung bình của các vị trí ở đây, các bạn xem lại nhé: Career path cho lập trình viên.

Ngoài ra, lập trình viên có rất nhiều cơ hội để **làm việc tại nước ngoài**. Trong các công ty lớn (FPT, Bosch, KMS), nhiều phòng ban thường có những đợt cử lập trình viên đi onsite qua nước ngoài (3 tháng tới 1 năm) để làm việc với đối tác Nhật, Úc, Anh.

Các bạn đam mê nước Nhật có thể rèn luyện tiếng Nhật và theo đuổi vị trí kĩ sư cầu nối – BrSE (Nên theo dõi blog về nghề BrSE rất hay của anh Tiger Nguyễn nhé).



Nếu có khả năng ngoại ngữ tốt, các bạn cũng có thể tự ứng cử, nộp CV cho các công ty nước ngoài. Một anh đồng nghiệp ngày xưa ở Aswig Solutions của mình từng tự xin việc và qua làm cho 1 công ty bên Singapore, bản thân thằng bạn thân mình cũng từng nhận được offer qua Nhật làm.

Tuổi nghề chỉ khoảng 30-40, có thật vậy không?

Nhiều người bảo rằng nghề lập trình tuổi nghề hơi thấp, sau một thời gian code thì đầu óc sẽ trở nên mất linh hoạt, không ngồi code lâu được, không học hỏi nhanh bằng giới trẻ.

Điều này cũng **có phần đúng ở Việt Nam**. Ở các công ty outsource, các công việc lặp lại nhiều, cường độ làm việc và OT cao nên làm lâu dễ ảnh hưởng đến sức khoẻ. Nếu cứ làm những công việc lặp đi lặp lại, không chịu cập nhật kiến thức mới thì bạn sẽ rất dễ bị lỗi thời và đào thải.

Tuy vậy, những người đi làm lâu cũng **có những lợi thế** nếu họ rành rẽ về cấu trúc hệ thống, qui trình làm việc cũng như kinh nghiệm lập trình. Những kiến thức này giúp họ **vươn lên tầm cao hơn** như làm PM, làm quản lý, hoặc lên tầm Software Architecture để thiết kế hệ thống.

Khi đi làm, bản thân mình và bạn bè vẫn thấy có những bác 4-50 tuổi vẫn **code khoẻ và sung hơn giới trẻ**, lâu lâu có công nghệ mới vẫn lao vào tìm hiểu như thường. Nếu đã có khả năng và đam mê thì bạn không cần lo lắn chuyện tuổi nghề v…v nhé.

Kết

Mình không biết chắc chắn 5-10 năm nữa ngành lập trình sẽ ra sao, nhưng theo suy đoán của mình thì **nhu cầu chỉ có tăng chứ không giảm** (Bằng chứng là các code camp, bootcamp ở nước ngoài vẫn đang mọc lên liên tục như nấm sau mưa) nên các bạn học chắc sẽ không lo thất nghiệp!

Tuy vậy, các bạn cũng đừng quên rằng điều quan trọng nhất không phải là công việc tốt, lương cao mà là… đam mê. **Phải có đam mê** thì bạn mới có thể theo đuổi được ngành này nhé!

Hai con đường trở thành lập trình viên: Đại Học và Học Đại

Ở chương trước, mình đã chia sẻ về triển vọng nghề nghiệp và lương bổng của ngành lập trình. Trong chương này, chúng ta cùng tìm hiểu về hai con đường để trở thành một lập trình viên: Đại Học và Học Đại (tự học), cùng với những **thuận lợi và khó khăn** khi lựa chọn chúng nhé.

Đại Học – Con đường dễ đi (Dù không bằng phẳng)

Trở thành lập trình viên bằng con đường Đại Học, nghĩa là thi đậu Đại Học và theo học ngành Công Nghệ Thông Tin (Khoa Học Máy Tính/Kĩ Sư Phần Mềm) của một trường Đại Học nào đấy.

Đây là con đường an toàn, khá dễ đi (thực ra học cũng hơi cực khổ chứ không quá dễ đâu) nên được nhiều bạn lựa chọn. Việc học Đại Học một cách chính qui có khá nhiều ưu điểm:

- **Vững kiến thức cơ bản**: Chương trình học của các trường ĐH thường được xây dựng một cách công phu, kĩ lưỡng. Sinh viên sẽ được học từ những môn lập trình cơ bản (C, C++) cho đến kiến thức nền tảng (cơ sở dữ liệu, thuật toán, hạ tầng mạng, hệ điều hành). Các kiến thức nền tảng này vô cùng quan trọng trong quá trình làm việc, học kiến thức mới.
- **Quan hệ**: Đi học, bạn sẽ được làm quen, học hỏi từ những người bạn có cùng đam mê, cùng sở thích. Khi có gì khó khăn, bạn có thể dễ dàng hỏi bạn bè hoặc thầy cô. Những mối quan hệ này rất có ích về sau này (Khi muốn tìm việc hay muốn học lên cao).
- **Cơ hội việc làm và thực tập**: Hầu hết các trường đều hỗ trợ hoặc hướng dẫn sinh viên đi thực tập. Đây là **cách tốt nhất để lấy kinh nghiệm**, trải nghiệm môi trường làm việc chuyên nghiệp, tạo lợi thế cho bạn khi đi xin việc.



Bằng Đại Học cũng khá là quan trọng đấy nhé!

Các bạn có thể xem thêm về những lợi ích mà học Đại Học mang lại trong bài: Lập trình viên có cần học Đại Học hay không?.

Tuy vậy, nếu đi con đường này, bạn cần lưu ý những khuyết điểm của nó:

- **Dễ bị thói quen ỷ lại**: Do chương trình học đã cố định, nhiều bạn cứ nghĩ học hết các môn trong trường là đã đủ kiến thức để đi làm. Điều này dẫn đến thái độ ỷ lại, không tự học mà chỉ đợi được dạy. Thái độ này vô cùng nguy hiểm khi đi làm.
- **Kiến thức cũ, không được cập nhật**: Kiến thức trong trường Đai Học chỉ là kiến thức cũ và cơ bản, không đủ để làm việc (Xem lại bài viết Những điều Đai Học không dạy bạn). Ngoài ra, ta còn phải học một số môn khá **nặng nề mà vô dụng** như: quân sự, triết học Mác Lê Minh, Lý Hoá Đại Cương,…
- **Tốn thời gian và tiền bạc**: Học Đại Học đồng nghĩa với việc bạn bỏ mất 4 năm thời gian để mài đít trên ghế nhà trường, đóng tiền nhà, tiền học phí v…v. Đây là một khoảng đầu tư khá lớn. Xét về mặt kinh doanh, học ĐH ra mà không kiếm được việc làm đồng nghĩa với việc bạn đầu tư… thua lỗ.



Ngoài ra, có thể bạn sẽ mất kha khá thời gian cho việc chơi game, gái gú ở thời Đai Học

Học Đại – Con đường gập ghềnh, lắm chông gai vất vả

Học Đại Học **không phải là con đường duy nhất** để trở thành lập trình viên. Có khá nhiều bạn trở thành lập trình viên bằng cách tự học ở trung tâm, tự học thêm trên mạng hoặc qua sách vở.

Đây là con đường dành cho các bạn không có thời gian hoặc điều kiện; hay đã tốt nghiệp ngành khác, có đam mê với công nghệ thông tin và **muốn tự học lập trình**.

So với học Đại Học, con đường này có lắm chông gai, lắm gian nan thử thách hơn nhiều:

- **Bối rối không biết hướng đi**: Rất nhiều bạn hỏi mình "Em muốn tự học lập trình, nhưng không biết bắt đầu từ đâu?". Thật vậy, **kiến thức trong ngành lập trình rất rộng** và vô cùng bao la. Các bạn tự học thường dễ "ngộp" vì lượng kiến thức khổng lồ và không biết bắt đầu như thế nào.

- **Dễ nản và bỏ cuộc**: Thật lòng mà nói, việc lập trình và học lập trình không hề dễ dàng. Bạn không thể thành thạo lập trình chỉ sau ngày một ngày hai, mà phải trải qua nguyên một quá trình học tập rèn luyện dài đằng đẳng. Quá trình dài dòng và gian khổ này dễ khiến nhiều bạn nản lòng và bỏ cuộc.
- **Hổng kiến thức căn bản**: Kiến thức căn bản chỉ có trong sách vở, lại khá nặng nề nên nhiều bạn tự học thường bỏ qua hoặc học sơ sài. Điều này dẫn đến việc nhiều bạn tự học lập trình bị mất căn bản, có thể sử dụng công nghệ nhưng không hiểu rõ bản chất của chúng.
- **Khó tìm việc hơn**: Mặc dù ngành IT có tiếng là "không quá coi trọng bằng cấp", rất nhiều công ty chỉ tuyển người đã tốt nghiệp. Bạn chưa tin à? Đây là một số mẫu tuyển dụng mình vừa hốt trên careerbuilder về:

Yêu Cầu Công Việc

- Good experience on Java core, Spring/Hibernate, J2EE
- Good knowledge and experience on OOP
- Familiar with one of the followings: Java CDI, JAX-WS, JDBC, MySQL
- Experience with Frontend: JQuery, basic HTML/HTML5 and CSS is preferred
- Good at English communication (Reading/Writing/Speaking/Listening)
- Self-motivated, detail oriented, client oriented

Education:

- Bachelor/Engineer Degree preferably in Computer Science/IT.
- Technician Diploma/Degree will be considered.

Yêu Cầu Công Việc

- Bachelor's degree in Computer Science, Mathematics, or other related scientific or technical discipline.
- At least 1+ years of application development experience.
- Proficiency in OOP, Java programming language, J2SE, and Swing.

Yêu Cầu Công Việc

- University degree in IT or Computer science
- Experienced with .NET and Java Technology
- Experienced in Web Development

Bạn có để ý là chúng đều đòi hỏi Bachelor/Engineer Degree, hoặc trình độ Đại Học ko? Không có bằng ĐH, cơ hội việc làm của bạn sẽ **hẹp hơn các bạn sinh viên** đã ra trường nhiều.

Tuy vậy, khi đi con đường này, bạn sẽ có được một số ưu thế sau:

- **Muốn học gì thì học**: Bạn có thể lựa chọn chỉ học những thứ mình thích, những thứ khiến mình hứng thú. Không cần phải phí thời gian nhồi vào đầu những môn đại cương, những kiến thức triết học vô bổ nữa.
- **Rèn được kĩ năng tự học**: Đây là một một trong những kĩ năng quan trọng nhất của lập trình viên. Biết cách tự học, bạn sẽ dễ dàng nắm vững công nghệ mới, đồng thời giữ cho kiến thức của mình không bị lạc hậu.

- **Tiết kiệm tiền bạc**: Tự học đồng nghĩa với việc bạn không phải lên trường, không phải đóng học phí, không tốn tiền mua sách vở học tập.
- **Dễ sắp xếp thời gian**: Bạn có thể tự học mọi lúc mọi nơi, dễ dàng thay đổi thời gian địa điểm học. Do đó, cách này khá phù hợp với những bạn đang đi làm, bận rộn, ít có thời gian rảnh.



Trong ngành IT, vẫn có khá nhiều lập trình viên giỏi, lương cao dù không hề được đào tạo một cách chính qui

Kết

Trong bài viết này, mình đã phân tích rõ hai con đường thường gặp để trở thành một lập trình viên. Cá nhân mình khuyên các bạn nên **lựa chọn con đường học Đại Học**. Dẫu có hơi mất thời gian nhưng nó khá là an toàn, ổn định và dễ đi hơn.

Tuy vậy, mình cũng biết có nhiều bạn vì hoàn cảnh, do đam mê nên phải lựa chọn con đường thứ hai – Học Đại. Vì thế, mình sẽ **dành nguyên chương** để chỉ dẫn kinh nghiệm, hướng đi và lộ trình học cho các bạn muốn tự học lập trình.

Con đường nào cho các bạn tự học lập trình?

Ở chương trước, mình đã nói về hai con đường để trở thành lập trình viên: Đại Học và Học Đại.

So với việc học Đại Học, con đường tự học – học đại có **nhiều thử thách và gian nan trắc trở** hơn nhiều. Do vậy, mình dành nguyên bài viết này để định hướng, chia sẻ về con đường dành cho các bạn muốn tự học lập trình. Hi vọng chúng sẽ có ích cho bạn.

Xác định lý do muốn học lập trình

Đầu tiên, phải xin cảnh báo trước với các bạn là việc học và việc lập trình **không hề dễ dàng**. Chuyện tự học lại càng khó khăn hơn và không phải ai cũng có thể theo đến cùng.

Kiến thức lập trình phức tạp, khó tiếp thu. Khối lượng kiến thức nhiều lại hay thay đổi. Đây là lý do mà nhiều bạn dễ cảm thấy nản lòng, muốn bỏ cuộc khi đang học.

Do đó, mình nghĩ trước tiên các bạn nên **xác định lý do** mình muốn học lập trình: vì đam mê với IT, học để khởi nghiệp, học để làm ra sản phẩm đổi đời. Khi có ý định bỏ cuộc, hãy nghĩ đến lý do tại sao mình bắt đầu học.



Ngoài ra, việc xác định lý do học sẽ giúp bạn dễ dàng **lựa chọn lộ trình học** hơn. VD bạn muốn làm web thì chỉ cần học về web, muốn làm app di động thì chỉ cần học kĩ về di động, không phải lan man học đại trà.

Lựa chọn một con đường để đi

Để có thể đi làm, lập trình ra một thứ gì đó, hầu như lập trình viên nào cũng trải qua những giai đoạn sau:

1. **Nhập môn**: Chọn một ngôn ngữ nào đó (C, C++, Python) để nhập môn, hiểu các khái niệm cơ bản trong lập trình (biến, hàm, con trỏ, module). Giai đoạn này mất khoảng 1-2 tháng.

2. **Nhập môn sâu hơn**: Học C++ hoặc Java/C# để tìm hiểu về các khái niệm OOP, về cấu trúc dữ liệu và thuật toán. Ngoài ra, bạn còn phải học về cách thiết kế database và cách chúng hoạt động. Giai đoạn này cũng mất khoảng 1-2 tháng.
3. Những giai đoạn về sau mất từ vài năm cho tới vài chục năm để thành thục.
4. **Chuyên sâu về ngôn ngữ**: Sau khi đã nắm các khái niệm cơ bản, các bạn bắt đầu tìm hiểu chuyên sâu một ngôn ngữ nào đó: Ngôn ngữ đó điểm mạnh điểm yếu gì, làm được những gì, cách thiết kế code, cách viết hàm ra sao. Bạn cũng phải tìm hiểu về hệ sinh thái của ngôn ngữ đó (C# thì đi với Window, MS SQL và VS, PHP thì đi với Linux, MySQL, ….)
5. **Kiến thức nâng cao**: Mỗi ngôn ngữ đều đi cùng với nhiều framework và thư viện. Phải có những kiến thức nâng cao này thì bạn mới có thể xin việc làm, làm được việc.
    1. Bạn chọn Java Web, bạn phải biết về Struts, Hibernate…
    2. Theo Android thì phải rành Java, hiểu rõ về LifeCycle của app Android, các khái niệm như Acitivty, Fragment,…
    3. Theo C# thì bạn phải biết ASP.NET MVC, Entity Framework…
6. **Kiến thức phụ thêm**: Kiến thức về cách dùng Git/SVN, về HTTP và AP Về cách viết code, về kiến trúc phần mềm, caching….



Mình khuyên các bạn tự học cũng **nên đi theo hướng tương tự**. Hãy xác định thứ mình muốn học rồi tìm tài liệu "nhập môn lập trình" để học những thứ cơ bản trước, sau đó học dần lên.

Điều quan trọng ở đây là các bạn phải rèn cho mình thói quen học tập, có thói quen, bạn sẽ không dễ dàng bỏ cuộc giữa chừng.

Tìm kiếm tài liệu và chọn cách học

Tài liệu học lập trình tiếng Việt khá ít và hơi lộn xộn, chủ yếu là bài tập. Do đó mình khuyên các bạn nên tìm tài liệu tiếng Anh để học, sau này cũng cần dùng tiếng Anh nhiều, học từ bây giờ sẽ tốt hơn.

Dưới đây là một số tài liệu dạng nhập môn cơ bản:

- JS For Cat (miễn phí): Giải thích các khái niệm lập trình một các vô cùng dễ hiểu (mèo cũng hiểu)
- The C Programming Language: Quyển sách kinh điển về lập trình C
- Learning to Program: Một cuốn sách khá hay cho các bạn nhập môn lập trình.
- Head First Programming: Nhập môn lập trình bằng ngôn ngữ Python. Sách rất hay với nhiều ví dụ hình ảnh minh hoạ sống động.

Ngoài ra, các bạn có thể học một số khoá free tại các trường dạy code online như khanacademy, udacity, codeacademy. Bonus: Đây là link 515 khoá học lập trình online **miễn phí** của các trường đại học danh giá https://medium.freecodecamp.com/515-free-online-programming-computer-science-courses-you-can-start-in-april-8b0ce1817d61

Bản thân mình **không khuyến khích** các bạn sưu tầm tài liệu. Dân VN mình có sở thích tải rất nhiều tài liệu, sách vở về nhưng… để đó không bao giờ đọc.



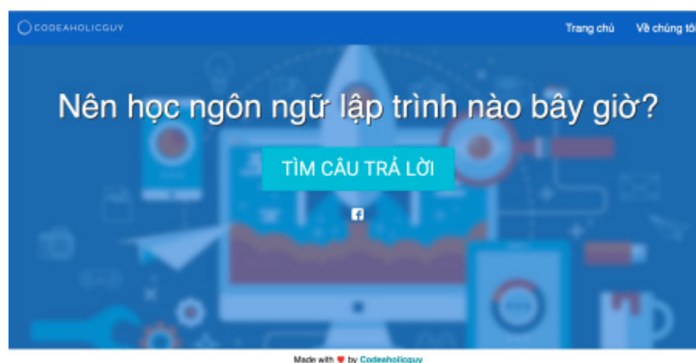
Tài liệu lập trình tải tùm lum nhưng không bao giờ đọc

Ở giai đoạn đầu, các bạn hãy chịu khó… làm bài tập. Việc ngồi làm bài tập, ngồi gõ code là cách hay nhất để các bạn nhớ syntax của một ngôn ngữ, nhớ cách gọi hàm, luyện thói quen viết code, tự học cách sửa các lỗi hay gặp.

Hoặc nếu lên một số trang như codeacademy, codeschool, các bạn sẽ được học một cách interactive. Tức là thay vì chỉ đọc và nghe, bạn sẽ được code từng bước theo hướng dẫn. Hệ thống sẽ chấm ngay cho bạn, báo cho bạn biết chỗ sai để mà sửa.

Cách học lập trình tốt nhất vẫn là … làm. Sau khi đã quen với một ngôn ngữ, việc làm bài tập sẽ **không giúp ích nhiều cho bạn nữa**. Để tăng tư duy lập trình, hãy tự đặt ra cho mình một dự án nho nhỏ: ứng dụng tính tiền hoặc quản lý thời gian, web bán hàng v…v.

Bạn sẽ thấy mình học được nhiều thứ hơn so với việc ngồi giải bài tập nhiều.



Làm một cái pet project nho nhỏ thế này này. Sử dụng tại: language.codeaholicguy.com

Lời kết

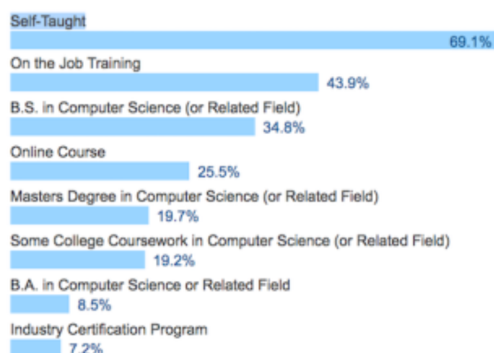
Điều cuối cùng mình muốn khuyên các bạn là: lên kế hoạch ít thôi, đừng quá phí thời gian cho việc lên kế hoạch và trì hoãn mà hãy **cắm đầu vào làm ngay** đi!

Sau khi đọc xong bài này, hãy lập tức làm theo 4 bước sau:

- Xác định mình muốn học gì, làm gì
- Xác định sơ lộ trình học
- Cắm đầu vào học, sau đó tìm tài liệu dần và cập nhật lại lộ trình
- Áp dụng kiến thức đã học vào để code ra một cái gì đó.

Khi đã thành thạo những thứ cơ bản, bạn hãy chuyên sâu vào một ngôn ngữ nào đó và bắt đầu dùng nó để làm các dự án nho nhỏ. Đừng quá quan tâm lo sẽ học sai ngôn ngữ, ngần ngừ đắn đo quá lâu, developer giỏi nào cũng biết vài ngôn ngữ.

Cứ đi đi, rồi sẽ thành đường thôi, lâu lâu có bị hơi lạc hướng thì cứ quay lại **làm theo kế hoạch do chính mình đề ra** nhé! Chúc các bạn may mắn.



Post tấm hình chia sẻ nè, 70% lập trình viên trên stackoverflow đều tự học đó

Học ngôn ngữ lập trình đầu tiên như thế nào?

Là lập trình viên thì dĩ nhiên là phải biết … lập trình. Tuy nhiên, một trong những khó khăn **khiến nhiều bạn bỏ cuộc**, đó là việc chọn và học ngôn ngữ lập trình đầu tiên.

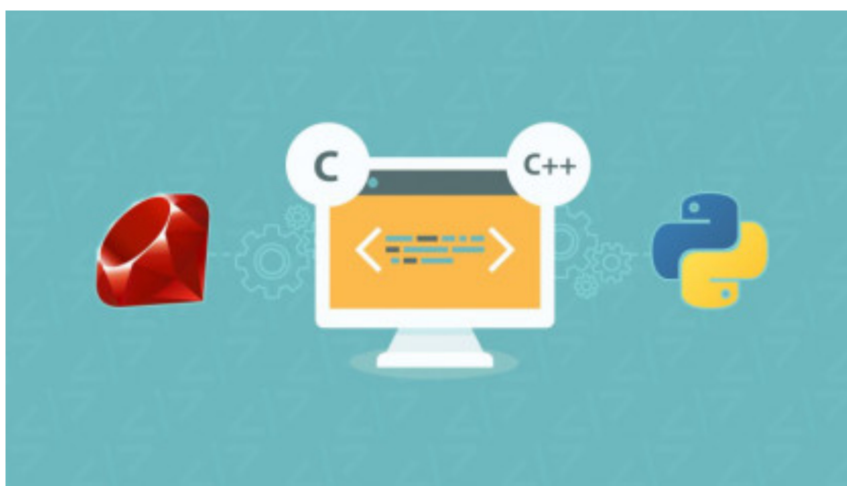
Dù cho bạn có học Đại Học, hay tự học lập trình, bạn đều sẽ phải đối diện với những khó khăn này! Vì vậy, mình sẽ chia sẻ **một số kinh nghiệm và định hướng** khi học những ngôn ngữ lập trình đầu tiên.

Đầu tiên nên học ngôn ngữ gì?

Mình hay nhận được câu hỏi từ nhiều bạn: Nên bắt đầu bằng ngôn ngữ lập trình gì? Thật sự, ở giai đoạn đầu, học ngôn ngữ gì không quan trọng như bạn nghĩ!

Tại sao vậy? Ở giai đoạn đầu, bạn học là để nắm cơ bản, để hiểu về lập trình, chứ không phải học rồi **theo ngôn ngữ đó cả đời**! Khi đã có trình độ, bạn có thể học một ngôn ngữ/công nghệ chỉ trong vài ngày hoặc vài tuần. Do vậy đừng quá lăn tăn về chuyện chọn ngôn ngữ. Thay vì ngồi đắn đo suy nghĩ, cứ bắt đầu học đi thôi!

Đầu tiên, bạn có thể **chọn một ngôn ngữ đơn giản** để nhập môn trước. Mình thấy đa phần các trường ở Việt Nam dạy **C**, đây là một ngôn ngữ khá hay, ngắn gọn, giúp bạn hiểu cách máy tính hoạt động. Một số trường nước ngoài dạy **Python**, cú pháp khá gọn và trong sáng, cũng khá thích hợp cho newbie.



Có thể chọn C hoặc Python làm ngôn ngữ lập trình đầu tiên

Sau đó, hãy học một ngôn ngữ lập trình nào đó **có hỗ trợ OOP**: C++, C#, Java. Hãy nhớ rằng ngôn ngữ không quan trọng, mục đích của bạn là để **làm quen và nắm vững các khái niệm OOP**.

Nguồn học ở đâu? Nếu tiếng Anh không giỏi, các bạn có thể lên Google tìm: Giáo trình C, Giáo trình Java v...v và tải giáo trình về học. Giáo trình ở các trường Đại Học Việt Nam tuy hơi cũ nhưng cũng khá hiệu quả, vì chúng chỉ tập trung vào các khái niệm cơ bản, syntax của ngôn ngữ. Nếu tiếng Anh kha khá, các bạn hãy lên các trường online để tập học dần từ bây giờ, sau này đi làm cũng phải học bằng tiếng Anh thôi hà.

Học những gì trong đấy?

Ở giai đoạn đầu, bạn sẽ cảm thấy khá khó khăn vì có quá nhiều khái niệm mới, quá nhiều điều cần học. Tuy nhiên, chúng thường bao gồm những điều sau đây:

- Cú pháp (syntax) ngôn ngữ
- Biến và con trỏ
- Cấu trúc điều kiện (if/else)
- Vòng lặp
- Hàm
- Đọc/ghi file
- Một số thư viện và hàm cơ bản

Sau khi học về OOP, bạn sẽ cần học thêm một số điều như:

- Bốn thuộc tính của OOP
- Lý do sử dụng OOP
- Class, Object, Modulde, Namespace
- Access Modifier
- Nguyên lý thiết kế OOP (Cái này khá khó, đi làm có nhiều kinh nghiệm đôi khi làm vẫn không đúng)
- Các khái niệm như cấu trúc dữ liệu và thuật toán cũng **khá là cần thiết** và khó học, nên mình sẽ nhắc đến chúng ở một bài viết khác.



Có thể học về OOP thông qua C++, Java hoặc C#. C++ hơi khó hơn một chút

Nắm vững khoảng 1,2 ngôn ngữ, các khái niệm OOP, là bạn đã có một nền tảng khá ổn trong ngành rồi đấy (Nói vậy chứ không dễ đâu nhé)!

Nếu có thời gian rảnh, bạn có thể đi sâu vào từng ngôn ngữ hoặc thư viện, vd như LINQ trong C#, Template trong C++, Generic trong Java, Reflection.... Các khái niệm về networking, multithread, concurrency control... cũng là những khái niệm khó và hay, ngôn ngữ nào cũng dùng được, hãy cố gắng nắm vững chúng.

Học sao cho code giỏi? Nản quá phải làm sao?

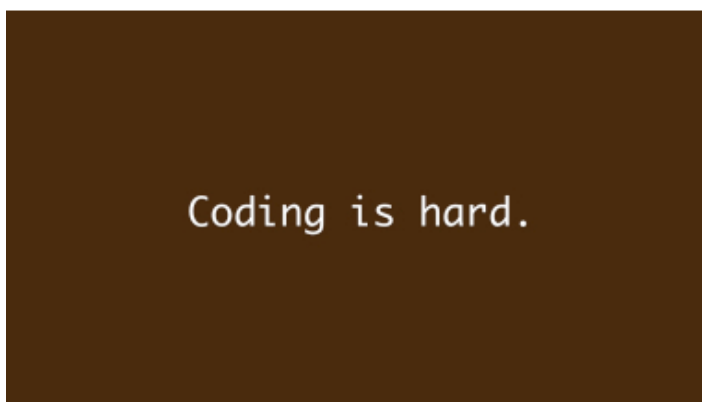
Như mình đã chia sẻ trong bài Căn bản và tư duy lập trình, muốn code giỏi thì các bạn phải … code nhiều. Ở giai đoạn đầu, các bạn hãy chịu khó **làm bài tập nhiều, viết code nhiều** để quen với cú pháp của ngôn ngữ, quen với cách tư duy.

Khi có lỗi thì nên tự sửa, đừng hễ gặp lỗi là mang lên các forum này nọ hỏi. Làm theo cẩm nang fix bug, hãy tập đọc message lỗi để tìm cách sửa các lỗi cú pháp, lỗi khi chương trình chạy sai. Chỉ cần luyện tập nhiều là bạn sẽ giỏi lên thôi!

Nhiều bạn chia sẻ với mình là cảm thấy code khó quá, học rất nản, sợ không theo ngành nổi. Đừng lo, hầu hết dân trong nghề đều công nhận là **code thật ra khá khó**.

Ngày xưa, khi ngồi làm bài tập, code mình viết toàn sai và bị lỗi. Mình cũng từng rất nản, **nghĩ rằng mình không hợp với lập trình** (mình chắc ai cũng từng cảm thấy như thế).

Tuy nhiên, sau khi code nhiều, tiếp xúc nhiều với lập trình, k**hả năng của bạn sẽ tiến bộ dần lên**. Lúc đó, những vấn đề phức tạp ngày trước không còn làm khó bạn được đâu. Cố lên nhé!



Code không hề dễ dàng, nên bạn chớ vội nản và bỏ cuộc!

Học xong thì làm gì?

Một số bạn có câu hỏi là **em đã học xong** ngôn ngữ này ngôn ngữ kia, giờ em nên làm gì?

Đầu tiên, **không có khái niệm học xong một ngôn ngữ**, chỉ có xong một môn trên trường mà thôi. Nhiều người đi làm 5-10 năm mà còn cảm thấy mình **không "xong" nổi** một ngôn ngữ. Kiến thức bạn học được trong trường rất ít, chưa đủ lên tầm junior nữa, đừng tự tin quá!



Học thế nào mới là "xong"??

Nếu đã khá vững một ngôn ngữ, bạn có thể tìm và tải source code của một ứng dụng/thư viện viết bằng ngôn ngữ đó. Xem cách tổ chức code thế nào, cách viết code, đặt tên biến ra sao. Đây là điều mà **các bạn mới ra trường** hay thiếu.

Lý do là ở trong trường, chúng ta chỉ cần **viết code cho chạy được** là xong. Tuy nhiên, thiết kế cấu trúc toàn bộ dự án ra sao, viết code thế nào cho dễ đọc, dễ hiểu, dễ bảo trì là những điều trường học không hề dạy.

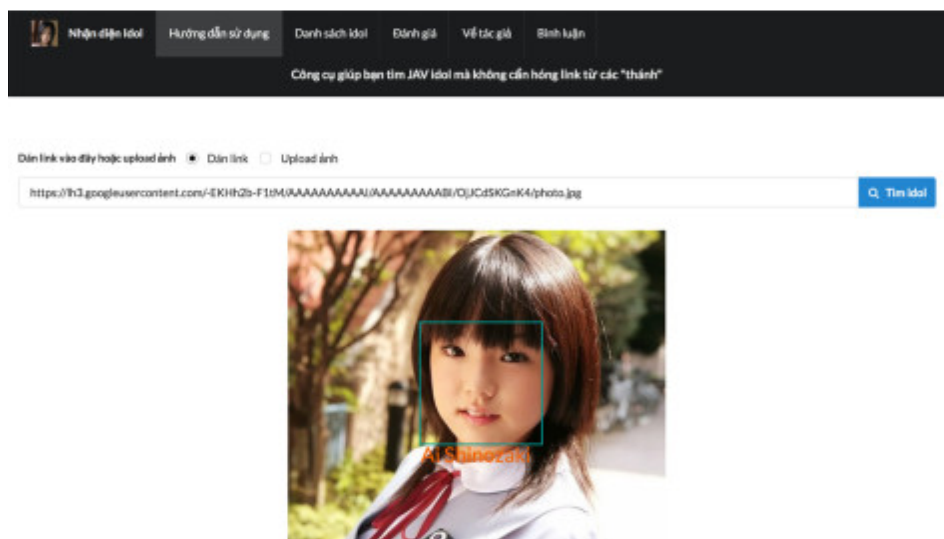
Do vậy, các bạn hãy chịu khó xem code người khác viết viết để lấp đầy khoảng trống kiến thức của mình nhé!

Học framework, học xong là phải làm!

Tiếp theo, hãy bắt đầu **học một framework nào đó!** Ví dụ bạn học *Java*, hãy thử tìm hiểu *Spring hoặc Android*. Học *C#* thì thử tìm hiểu *ASP.NET MVC* hoặc *WPF*.

Kiến thức và kinh nghiệm sử dụng framework là thứ mà các công ty đang cần. Nhưng nhớ là phải nắm căn bản trước khi ham hố nhảy vào học framework nhé!

Tới một lúc nào đó, việc làm bài tập sẽ **không có ích gì nữa**. Lý thuyết phải đi đôi với thực hành. Lúc này, việc bạn cần là tạo ra một sản phẩm. Không cần phải là một thứ gì to tát, hãy áp dụng kiến thức mình đã học để làm cái gì đó nho nhỏ đơn giản (Xem các gợi ý trong bài viết về pet project nhé).



Học xong là phải LÀM. Làm một vài ứng dụng có ích, nho nhỏ như Nhận Diện JAV Idol cũng được!

Vững kiến thức cơ bản, biết cách dùng framework, có dự án trên GitHub; những điều đó khá là đủ cho bạn **có một CV khá đẹp** để đi xin thực tập. Nếu có thể, hãy ráng học một số kĩ năng tìm việc, sau đó xin đi thực tập từ năm 3 năm 4 nhé! Không có cách học nào nhanh bằng việc học trong môi trường thực tế, tiếp xúc với dự án thật đâu!

Về kinh nghiệm thực tập và phỏng vấn, các bạn xem lại trong series muôn nẻo đường tìm việc nhé!

Kết

Như mình đã nói, việc học ngôn ngữ lập trình đầu tiên **không quá quan trọng,** cũng **không quá khó khăn** như bạn nghĩ. Vấn đề là liệu bạn có biết tự **định hướng bản thân**, có dám **bắt tay vào học và làm ngay** hay không thôi!

Năm con đường kiếm tiền từ nghề lập trình

Một trong những câu hỏi mình hay nhận được về ngành IT là "ngành này ra trường có dễ xin việc, có dễ kiếm tiền không"? Câu trả lời **dĩ nhiên là CÓ!** Là lập trình viên, có **rất nhiều cách** để bạn kiếm tiền. Trong chương này, chúng ta cùng tìm hiểu 5 con đường để kiếm tiền từ ngành này nhé!

Freelance hoặc code theo hợp đồng

Kiếm tiền bằng cách lập trình freelance, hoặc code theo hợp đồng, là cách kiếm tiền **khá dễ thực hiện**, phù hợp với các bạn sinh viên hoặc những người đã đi làm.

Bạn có thể tìm dự án trên các trang freelance trong vào ngoài nước như: vLance.vn, freelancer.com, upwork.com. Các công việc này thường là code web, code app mobile,… với **đủ mọi ngôn ngữ và công nghệ**.

Nếu có quan hệ, bạn cũng có thể **tự kiếm một số hợp đồng** với các doanh nghiệp, cá nhân để code app hoặc web kiếm tiền. Ví dụ thấy bà bán xôi ngoài ngõ, bạn có thể dụ dỗ code cho bà cái web bán xôi online, tính giá 1-2 triệu chẳng hạn.



Ưu điểm của cách làm này:

- Bạn tự chủ được thời gian làm việc, thích thì làm không thích thì nghỉ, không cần nhìn mặt sếp.
- Nếu có uy tín, biết cách deal giá, đôi khi thu nhập còn cao hơn đi làm.
- Bạn cũng có thể vừa học vừa làm, không cần chờ tới lúc ra trường (Thời xưa mình học FPT từng nghe có vài bạn sinh viên năm 2-3 làm freelance đã kiến 2-30 củ mỗi tháng)

Nhược điểm của cách làm này cũng khá nhiều:

- Thu nhập **không ổn định**, nhiều khi phải căng mặt ra mà tìm job

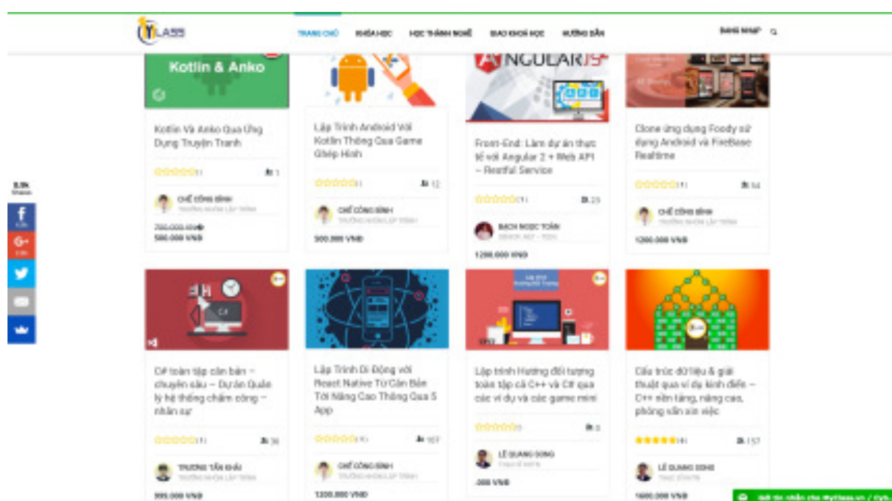
- Mặc dù bạn học hỏi được 1 số thứ (làm việc với khách hàng, lấy yêu cầu, học công nghệ yêu cầu v…v), bạn sẽ ít phát triển được khả năng kĩ thuật của bản thân (làm việc không có qui trình, không có team, không có người góp ý code)
- Đôi khi còn bị khách hàng kì kèo, bị huỷ hợp đồng, xù tiền
- Ngoài ra, bạn còn phải cạnh tranh với các developer Ấn Độ, Philippine, thậm chí là… Việt Nam. Do nước ta và các nước này đều nghèo nên dân tình đua nhau… hạ giá rất rẻ mạt, công việc làm ăn cũng khá khó khăn.

Tạo ra sản phẩm: web, app di động, khoá học

Một phương pháp kiếm tiền khác là tạo ra sản phẩm để… bán. Do đa phần việc tạo ra một sản phẩm website khá là **phức tạp, mất thời gian** nên thông thường thường là các công ty mới có thể đầu tư làm.

Về phần developer cá nhân, chúng ta có thể làm app di động (game), sau đó đăng lên store để kiếm tiền (tiền quảng cáo, bán app, bán sản phẩm trong game).

Ngoài ra, nếu có khả năng, bạn cũng có thể **tạo ra khoá học và đem bán**. Điển hình như ở nước ngoài, có bác John Sonmez từng kiếm được **500 nghìn đô/năm** nhờ các khoá học trên pluralsight. Ở Việt Nam, bạn có thể bán trên các trang như: myclass.vn, techmaster.vn, edumall.vn.



Một số khoá học trên myclass.vn

Ưu điểm của cách này là có thể **rèn luyện kĩ năng cá nhân**, có sản phẩm để show ra khi xin việc. Nếu may mắn, tiền thu được từ sản phẩm sẽ trở thành **thu nhập thụ động** (Tức là bạn chỉ ngồi không, **không phải làm việc mà vẫn có tiền** hàng tháng, sướng chưa).

Nhược điểm là cách làm này cũng **khá bấp bênh và hên xui**. Có vô số ứng dụng trên Android, iOS nên việc cạnh tranh khá gay gắt. May mắn như Nguyễn Hà Đông chỉ là thiểu số (Bằng chứng là sau Flappy Bird không thấy anh có game nào hot như vậy nữa).

Khoá học cũng vậy, có thể bạn sẽ bỏ ra rất nhiều công sức, nhưng cuối cùng **không bán được** vì… không biết marketing nên chả ma nào mua.

Làm startup hoặc mở công ty riêng

Hai tấm gương học IT xong ra khởi nghiệp thành công, lừng lẫy thế giới là Mark Zuckerberg và Bill Gates. Hiện tại, ở Việt Nam, có rất nhiều bạn tốt nghiệp IT ra trường và khởi nghiệp ngay.

Ưu điểm thì khỏi cần nói rồi. Tham gia startup, bạn sẽ học hỏi được vô số những điều bổ ích, phát triển bản thân. Nếu startup thành công, bạn vừa có tiền, có quyền lại có tiếng tăm. Chưa kể, cảm giác tự xây dựng được một sự nghiệp cho bản thân cũng **rất đáng tự hào** nhé.

Tuy nhiên, con đường khởi nghiệp và mở công ty có **khá nhiều chông gai vất vả**: đủ thứ sức ép và áp lực từ nhiều phía, tài chính khó khăn, gia đình bạn bè không thông cảm. Mấy thứ này báo đài đã nói nhiều lần nên các bạn cứ Google là ra nhe.



Hai tấm gương bỏ học và khởi nghiệp thành công,
còn mấy tấm gương thất bại thì giờ chìm đâu đó rồi

Đi làm ở công ty

Tốt nghiệp, ra trường, xin việc, đi làm, thăng tiến là con đường được rất nhiều người lựa chọn.

Đây là một con đường **khá an toàn, ổn định**. Hàng tháng bạn không phải lo đi tìm việc mà vẫn có việc làm, có lương. Mức lương của ngành IT cũng **khá cao so với mặt bằng chung**, nếu biết cách vun vén cũng đủ sống. Khi đi làm, bạn sẽ học hỏi được nhiều điều từ đồng nghiệp và sếp, từ qui trình làm việc của công ty.

Bạn cũng có thể **phát triển bản thân** lên các vị trí cao hơn, nhiều trách nhiệm hơn, dĩ nhiên lương cũng.. ngon hơn. Các bạn có thể đọc bài này về con đường phát triển nghề nghiệp của ngành IT.

Nhược điểm khi lựa chọn con đường này là bạn phải chịu khó đầu tư kĩ năng tìm việc (viết CV, phỏng vấn). Công việc cũng khá gò bó, đi làm 8 tiếng mỗi ngày, đôi khi phải OT để kịp tiến độ. Ngoài ra, nhiều khi phải **làm việc với đồng nghiệp, sếp dở hơi** cũng phải… cắn răng chịu đựng vì yêu cầu công việc.



Đôi khi cũng gặp phải sếp hoặc đồng nghiệp dở hơi

Kết hợp nhiều phương thức với nhau

Tất nhiên, nếu muốn kiếm nhiều tiền, bạn có thể **kết hợp nhiều phương thức** lại với nhau. Ví dụ, sáng đi làm, tối về cày freelance hoặc làm khoá học để kiếm thêm. Đôi khi sẽ hơi stress và mệt mỏi tí, tuy nhiên thu nhập cũng khá là cao.

Hoặc có nhiều bạn **chịu khó đi làm vài năm** để học hỏi. Sau vài năm, khi đã có vốn, có kinh nghiệm và quan hệ, họ mới bắt đầu mở công ty riêng. Cách này có phần **ổn định hơn** so với việc mở công ty ngay. Tuy nhiên, cũng cần phải học thêm khá nhiều, vì đôi khi điều hành một công ty **cần những skill hoàn toàn khác** so với lúc làm việc.

Kết

Qua bài viết, mình đã chia sẻ 5 con đường kiếm tiền từ ngành lập trình. Bản thân mình thì đang **áp dụng cách 2 và 4**: vừa đi làm, vừa dự tính làm sản phẩm để bán kiếm thêm. Nếu còn đang phân vân, không biết thuyết phục bố mẹ ra sao khi chọn ngành IT, hãy đưa họ xem bài viết này nhé.

Học "xong" lập trình thì làm gì, khi nào đi làm được??

Đây là phần cuối cùng của series "Nhập Môn Lập Trình Không Code". Trong bài viết này, mình sẽ giúp các bạn định hướng việc làm và giải đáp những thắc mắc như:

- Học lập trình tới khi nào là xong, có thể đi làm được?
- Làm sao vượt qua các vòng xin việc và phỏng vấn? Cần chú ý điều gì?
- Tại sao ra trường, có việc làm rồi vẫn phải tiếp tục tự học? Tự học như thế nào cho hiệu quả?

Bắt đầu thôi nào!

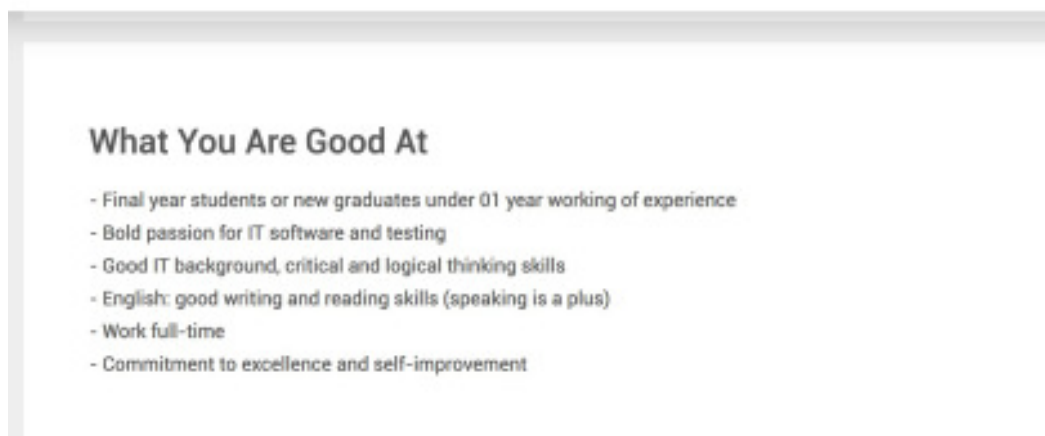
Học tới khi nào là xong, có thể đi làm được?

Đây là một câu hỏi mình được nghe rất nhiều lần, từ các bạn sinh viên lẫn những bạn tự học.

Câu trả lời là: học tới khi nào bạn **có đủ kiến thức** để đi xin thực tập, qua được vòng phỏng vấn:

- Với các bạn sinh viên, thường khoảng **cuối năm 2 hoặc năm 3** là các bạn đã đủ khả năng để đi xin thực tập (tất nhiên còn tuỳ ngộ tính: liệu bạn có học hành chăm chỉ không, có chịu khó tự học thêm không).
- Với các bạn tự học thì việc học tới lúc đủ cũng hơi … khó xác định. Nếu có lộ trình rõ ràng, học tập đều đặn **sau 6 tháng đến 1 năm** là bạn đã có đủ kiến thức để xin việc.

Nói vậy chứ kiến thức trong ngành rất rộng, khó mà biết đâu là đủ. Vì vậy, cách đơn giản nhất là bạn chịu khó lên các trang như itviec, vietnamwork đọc các mẩu quảng cáo tuyển dụng junior, fresher để xem họ đòi yêu cầu gì, sau đó tự tìm hiểu thêm.



Một mẩu tuyển dụng vị trí fresher trên Vietnamwork

Thường những kĩ năng họ yêu cầu chỉ là:

- Kiến thức lập trình cơ bản, OOP cơ bản
- Kiến thức cơ bản về ngôn ngữ: C#, Java
- Kĩ năng viết code, thể hiện bằng thuật toán
- Tiếng Anh đọc viết
- Biết sơ về framework và có project đã làm là lợi thế

Nếu đọc tin tuyển dụng, thấy những thứ họ yêu cần **mình đều hiểu đều biết** là bạn có thể tự tin đi xin việc làm rồi đó!

Làm sao để đi xin việc, đi phỏng vấn, xin thực tập

Tất nhiên, giỏi kĩ thuật **chưa chắc đã xin được việc.** Nếu bạn không biết cách thể hiện những gì mình có trên CV, không biết cách trả lời phỏng vấn, bạn sẽ khó tìm được công việc ưng ý!

Những vấn đề như: tìm việc ở đâu, viết CV ra sao, vượt qua kì phỏng vấn như thế nào… đều được mình đã chia sẻ trên blog hết rồi. Các bạn đọc lại 2 bài viết sau nhé:



Không phải cứ học trên trường là đủ, mà phải biết học sao cho không thất nghiệp

Một điểm nữa các bạn nên lưu ý là ở Việt Nam, các công ty thường cho **thực tập không lương, hoặc trả lương mang tính tượng trưng** khoảng 2-3 triệu. Công ty **có lý do để làm như vậy,** vì bạn chưa mang lại giá trị cho công ty, họ còn phải tốn công đào tạo bạn. Đừng quá quan tâm đến lương, mà hãy **tranh thủ học hỏi thật nhiều** trong giai đoạn này nhé!

Tuy nhiên, nếu sau 6 tháng đến 1 năm, mức lương của bạn vẫn bèo bọt hoặc miễn phí, nghĩa là công ty đang tìm cách bóc lột bạn đấy! Nếu cảm thấy mình có khả năng, hãy **ứng tuyển vị trí chính thức** với mức lương chính thức tại công ty; hoặc **tìm đường ra ngoài**

phỏng vấn nhé! Lúc này bạn đã có kinh nghiệm làm việc, dễ phỏng vấn và đòi lương hơn nhiều.



Đừng quá quan tâm đến lương, nhưng cũng nên cẩn thận kẻo bị "bóc lột" nhé

Học là chuyện cả đời, làm sao bồi dưỡng kĩ năng cứng, mềm?

Một điều mà mình đã **nhắc đi nhắc lại rất nhiều lần** trên blog là:

Trong ngành IT, bạn phải luôn luôn học hỏi và tiếp xúc với cái mới, do đó kĩ năng tự học là cực kỳ quan trọng.

Thông thường, có bốn lý do mà bạn phải tự học:

- **Do yêu cầu bắt buộc của dự án**: Ví dụ hồi xưa mình chỉ biết ASP.NET MVC, do phải là dự án WPF nên mình phải học WPF.
- **Học để dự phòng**: Có thể trong tương lai công ty sẽ phát triển thêm 1 mảng nào mới, hoặc nhận thêm một dự án mới. Việc học sẵn để dự phòng sẽ giúp bạn **có lợi thế hơn** so với các developer khác.
- **Học để theo kịp thời đại**: Công nghệ thay đổi liên tục, và nhiều công nghệ sẽ bị lỗi thời qua thời gian. Do vậy, bạn nên học để theo kịp bước tiến công nghệ. Dù có bỏ công việc hiện tại thì bạn vẫn có đủ skill để đi xin việc nơi khác.
- **Học để phát triển bản thân, để giỏi và có giá hơn**: Để trở nên có giá hơn, bạn phải giỏi cả technical lẫn kĩ năng mềm. Không có con đường nào khác ngoài việc học: học qua sách vở, qua blog, qua đồng nghiệp và sếp.



Đã chấp nhận theo nghiệp này là phải chịu khó học, học đủ thứ!

Ở Việt Nam, đa phần các bạn sinh viên còn chưa có thói quen tự học, mà hay dựa dẫm vào người khác, hoặc **chờ được dạy rồi mới học**. Do đó mình chia sẻ một số phương pháp tự hoc nhé:

- Đọc blog, đọc sách liên quan tới nội dung mình muốn học
- Xem các khoá học online, làm theo demo
- Tự áp dụng kiến thức đã học để làm một sản phẩm gì đó

Tất cả những điều này đều được mình đề cập qua hai bài viết: tiếp cận ngôn ngữ, công nghệ mới và làm pet project nhé.

Khoảng trống kiến thức giữa sinh viên IT và Lập Trình Viên

Do ngành IT đang dần thành một ngành hot ở Việt Nam (việc nhẹ lương cao, nhu cầu tuyển dụng nhiều), nhiều bạn sinh viên đổ xô vào chọn học các ngành công nghệ thông tin.

Tuy nhiên, các bạn sinh viên mới ra trường lại **dễ gặp phải tình trạng thất nghiệp**, hoặc khó kiếm việc làm. Nguyên nhân là do đâu?? Chẳng phải trường Đại Học nào cũng bảo "Vào trường tao học xong ra trường làm IT là lương ngàn đô" đấy sao?

Nguyên nhân cơ bản nhất là kiến thức được dạy trong trường **không bao giờ đủ**. Có một **khoảng trống lớn về kiến thức** giữa sinh viên IT mới ra trường và một lập trình viên "thực thụ".



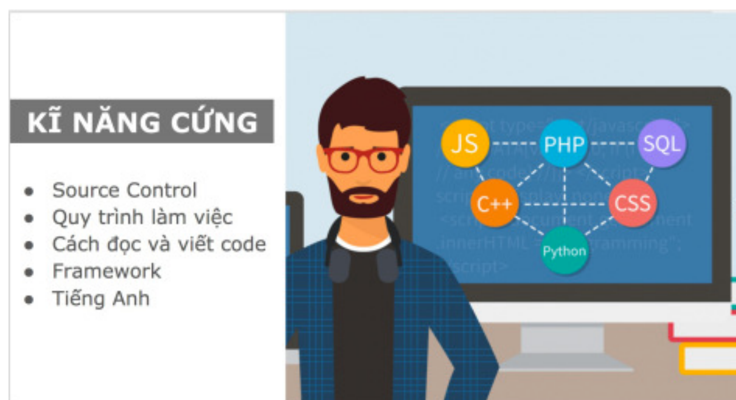
Trong chương này, chúng ta sẽ tìm hiểu và tìm cách lấp đầy khoảng trống này nhé.

Những thứ các bạn sinh viên thiếu bao gồm:

- Kĩ năng cứng
- Kĩ năng xin việc
- Kĩ năng mềm

Kĩ năng cứng

Những kĩ năng cứng này không có gì quá cao siêu như thuật toán, data mining, IoT, … mà chỉ là những thứ đơn giản như sau:



- **Source Control**: Thay vì viết code theo kiểu "thân ai nấy lo", các công ty sử dụng Source Control để quản lý toàn bộ source. Hai source control phổ biến nhất là Git và SVN (Một số công ty dùng TFS của Microsoft). Các bạn có thể tự tìm hiểu về Git, SVN rất dễ dàng bằng cách google git tutorial.
- **Quy trình làm việc**: Hiện nay, đa phần các công ty áp dụng qui trình Scrum, dựa trên các nguyên lý Agile. Thay vì đợi vào công ty mới học, các bạn có thể lên Scrum Training để tự học thông qua các video rất bổ ích. (Hoặc trong sách Code Dạo Ký Sự cũng có 2 bài viết rất cụ thể về Scrum và Agile).
- **Cách đọc và viết code**: Khi đi làm, phần lớn công việc của bạn sẽ là **bảo trì dự án**, viết thêm chức năng nên kĩ năng đọc code là hết sức quan trọng. Ngoài ra, ta bắt buộc phải **viết code rõ ràng, dễ đọc dễ hiểu** dễ bảo trì. Những điều này không thể học được ngày một ngày hai, mà phải trải qua quá trình rèn luyện. (Các bạn có thể tìm đọc thử Clean Code để tìm hiểu thế nào là code sạch).
- **Framework**: Trường đại học chỉ dạy cho bạn những kiến thức cơ bản. Nhưng bản thân các công ty lại yêu cầu sinh viên phải có kiến thức, kinh nghiệm về framework. Do đó, đừng chỉ mải mê học ngôn ngữ mà hãy chọn framework nào hay ho để học nhé.
- **Tiếng Anh**: Yếu ngoại ngữ sẽ làm bạn vụt mất rất nhiều cơ hội, vì tiếng Anh khá quan trọng trong ngành IT. Bạn cần tiếng Anh để đọc hiểu tài liệu, tự học, giao tiếp trao đổi với khách hàng. Mức lương cho các lập trình viên giỏi tiếng Anh dĩ nhiên cũng sẽ nhỉnh hơn nhé.

Kĩ năng xin việc

Có kĩ năng tốt nhưng thiếu kĩ năng xin việc thì dĩ nhiên là bạn cũng sẽ… thất nghiệp. Các kĩ năng xin việc cơ bản cũng khá đơn giản:



- **Tìm hiểu thị trường**: Mình đã nhiều lần nghe câu hỏi "Học ngôn ngữ lập trình gì bây giờ? Muốn biết câu hỏi này, bạn cần đọc các mẫu thống kê của itviec, vietnamwork để biết ngôn ngữ/công nghệ gì đang hot nhất (Gợi ý nhé, JavaScript giờ đang hot lắm đấy)
- **Định giá bản thân**: Muốn biết mức lương mà mình xứng đáng được nhận là bao nhiêu, đầu tiên các bạn cần xác định những kiến thức mình biết và số năm kinh nghiệm. Sau

đó, lên những trang như itviec, vietnamwork để **tham khảo mức lương trung bình** cho vị trí đó là bao nhiêu nhé.

- **Viết CV và Trả lời phỏng vấn**: CV là thứ giúp nhà tuyển dụng biết về bạn, giúp bạn vào vòng phỏng vấn. Để biết cách viết CV cho chuẩn, cách chuẩn bị và trả lời phỏng vấn, hãy xem lại series Muôn nẻo đường tìm việc trên blog nhé.

Kĩ năng mềm

Nếu kĩ năng cứng, kĩ năng xin việc quyết định chuyện bạn có việc làm hay không; kĩ năng mềm sẽ quyết định **khả năng sống sót** với nghề, **khả năng thăng tiến** của bạn.



- **Tự học**: Kiến thức trong ngành IT liên tục thay đổi. Các công nghệ mới liên tục lỗi thời, nếu không biết cách tự cập nhật kiến thức cho bản thân, bạn sẽ dễ trở nên **lỗi thời và lạc hậu,** khó cạnh tranh lại với lớp trẻ.
- **Làm việc nhóm**: Không ai code một mình! Trong phần lớn các dự án, bạn phải làm việc chung với đồng đội, với tester, với cấp trên. Để làm việc nhóm tốt, bạn cần kĩ năng giao tiếp, trình bày, giải quyết xung đột, những kĩ năng này sẽ **rất cần thiết** khi bạn muốn tiến lên những vị trí cao hơn.
- **Xây dựng tiếng tăm và quan hệ**: Biết cách xây dựng tiếng tăm, bạn sẽ nhận được sự nể trọng của đồng nghiệp cũng như cấp trên. Biết cách xây dựng quan hệ, bạn sẽ mở ra được rất nhiều những cơ hội mới (Bản thân mình có được việc làm ở Aswig và phòng IT của Lancaster cũng là nhờ bạn bè giới thiệu).
- **Chia sẻ kiến thức**: Chia sẻ kiến thức mang lại cho bạn rất nhiều lợi ích: Củng cố lại kiến thức, mở rộng mối quan hệ, xây dựng thương hiệu cá nhân. Hãy cùng tìm hiểu thêm về lợi ích của việc viết blog nhé.

Hi vọng qua bài viết này, các bạn sinh viên có thể nhận ra được những lỗ hổng kiến thức của mình, đồng thời tự tìm hiểu và lấp đầy những khoảng trống này nhé. Cứ thoải mái comment nếu bạn có câu hỏi cần giải đáp nha.

Bonus: Các bạn có thể tìm hiểu thêm qua slide thuyết trình và video phần trình bày của mình về chủ đề "Từ sinh viên IT tới Lập Trình Viên" nhé. Nếu chưa follow fanpage thì nhớ like và follow để đọc những bài viết hay mỗi ngày nghen.



Sinh viên IT học và làm gì để không thất nghiệp?

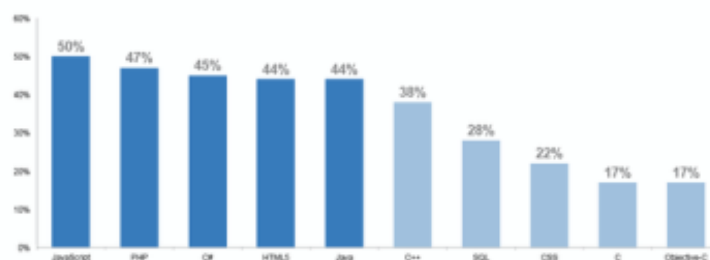
Ở chương trước, mình có chia sẻ về khoảng trống kiến thức giữa sinh viên IT và lập trình viên. Đó là một trong những lý do mà tuy ngành IT hiện đang khát nhân lực, các công ty tuyển dụng rất nhiều nhưng không tìm được người phù hợp.

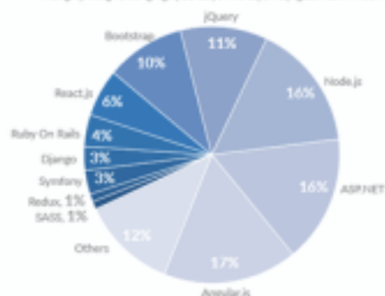
Là sinh viên, đã bao giờ bạn tự hỏi: **Mình phải học gì, làm gì để không thất nghiệp, ra trường có việc làm** chưa? Nếu chưa thì hãy bắt đầu tự hỏi từ bây giờ đi nhé! Bài viết này sẽ giúp bạn trả lời một phần câu hỏi trên.

Học cách tìm hiểu thị trường

Ngày xưa, các cụ có câu "Biết người biết ta, trăm trận trăm thắng". Muốn xin được việc thì phải **biết cách tìm hiểu thị trường**, xem ngôn ngữ nào đang hot, công nghệ nào đang được dùng nhiều để từ đó bổ sung kiến thức cho phù hợp.

Tìm hiểu thị trường có khó lắm không? Thật ra rất đơn giản, bạn chỉ việc chịu khó đọc một số khảo sát của các trang chuyên về việc làm như khảo sát sau: Khảo sát của topIT.





Đọc sơ qua, ta dễ thấy là JavaScript đang hot, tiếp đến là PHP, C#, Java. Về mảng web, các framework đang được dùng nhiều là Bootstrap, jQuery, AngularJS, NodeJS. Tự học các framework này sẽ giúp bạn dễ tìm việc hơn rất nhiều!

Để rõ ràng hơn, các bạn có thể xem một số mẫu tin tuyển dụng trên itviec, vietnamwork, careerbuilder để xem các công ty họ cần tuyển người có những kĩ năng gì, từ đó trau dồi thôi!

Yêu Cầu Công Việc

- Nắm vững Java Core và J2EE, có kinh nghiệm làm các ứng dụng Java Web.
- Có kinh nghiệm với Struts, Spring, Hibernate.
- Thao tác tốt với HTML, CSS, JavaScript (Jquery, AngularJS, Ajax, ...), Bootstrap, JSON, XML, ...
- Có hiểu biết về DI, OOP, MVC Framework, ORM, RESTful, Design Patterns.
- Thành thạo các công cụ quản lý mã nguồn: SVN, GIT, Ant, Maven, ...
- Sử dụng tốt các IDEs: Eclipse, NetBeans, ...
- Có kinh nghiệm làm việc với một trong các WebServer: Apache, Nginx, Tomcat, Jboss, Websphere ...
- Thao tác tốt với một trong các Database: MySQL, PostgreSQL, ... có kinh nghiệm làm việc với Big Data là một lợi thế.
- Có kinh nghiệm với Lucene, Solr, elasticsearch.
- Có kinh nghiệm với Google, YouTube & Facebook API.
- Có khả năng làm phân tích thiết kế database, kiến trúc hệ thống.
- Có khả năng làm việc tốt trong môi trường teamwork cũng như làm việc độc lập.
- Có thể làm việc dưới áp lực cao về deadline cũng như chất lượng sản phẩm.
- Khả năng tư duy tốt, chủ động trong công việc, có tinh thần trách nhiệm cao để hoàn thành công việc được giao.
- Sử dụng thành thạo tiếng Anh đọc và viết.

Một mẫu tuyển dụng Senior Dev Java.
Bạn thấy đấy, họ không chỉ yêu cầu ngôn ngữ
mà còn phải biết framework, design pattern, kĩ năng mềm

TUYỂN NHÂN VIÊN IT, LƯƠNG 5->10tr tuỳ năng lực
YÊU CẦU
- Biết quản trị web
- Biết quản lý Facebook profile, fanpage, group
- Biết quản lý kênh Youtube
- Biết xài Photoshop, Corel Draw ... và có thể tự học cách xài các phần mềm khác.
- Hồ sơ xin việc gồm: bản sao CMND có công chứng, bản sao hộ khẩu, sơ yếu lý lịch, KHÔNG CẦN BẰNG CẤP sẽ đánh giá trên năng lực
CÔNG VIỆC
- Các công việc liên quan đến IT, sẽ trao đổi cụ thể lúc phỏng vấn
- Làm việc toàn thời gian 8h30-18h30
QUYỀN LỢI
- Tuần nghỉ 1 ngày
- Lương + thưởng
- Nếu muốn làm ca, hãy liên hệ quản lý để thoả thuận
- Ca 1: 9h - 15h
- Ca 2: 13h - 19h
- Làm việc tại
LIÊN HỆ:
- Inbox FB hoặc ĐT

Các bạn cũng nên né những mẩu tuyển dụng dạng này ra nhé.
Vào đó là thành IT culi chứ không tăng kĩ năng code đâu!

Điều quan trọng nhất, nếu muốn có việc làm, đương nhiên là phải học cách… xin việc. Dù bạn có giỏi cỡ nào mà **không xin được việc** thì cũng… vô dụng. Muốn xin được việc phải viết được một mẫu CV ổn, phải qua được vòng phỏng vấn.

Vậy làm sao để viết CV đẹp, làm sao vượt qua vòng phỏng vấn tuyển dụng? Mình đã có viết đầy đủ rồi nên các bạn chịu khó xem lại series Muôn nẻo đường tìm việc nha.

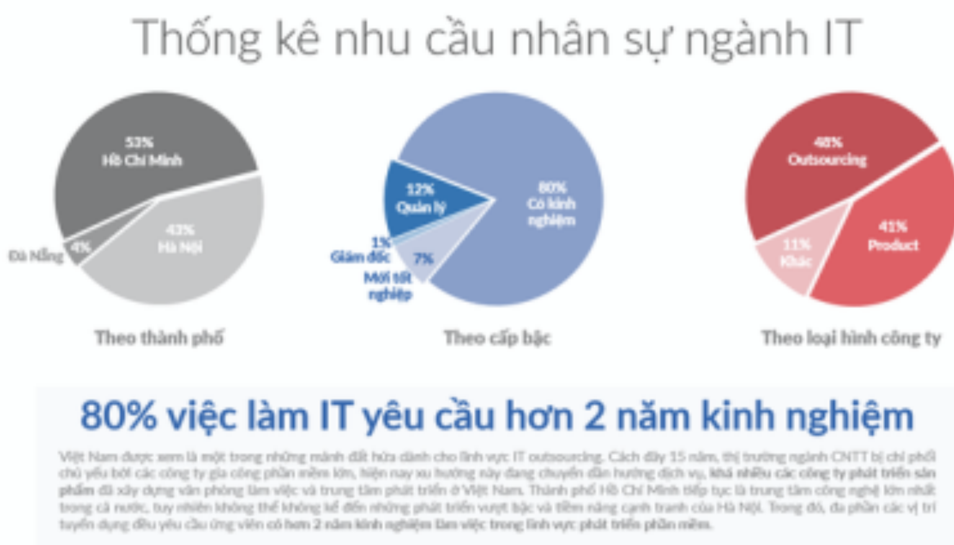
Lấy kinh nghiệm ở đâu ra?

Có một thực tế trớ trêu như thế này: Nhiều công ty chỉ thích **tuyển người đã có kinh nghiệm**!

Sinh viên ra trường không có kinh nghiệm -> thất nghiệp -> không được làm việc -> không có kinh nghiệm, tạo thành một vòng luẩn quẩn.



Nghe đồn có công ty tuyển sinh viên vừa tốt nghiệp nhưng đòi 6 năm kinh nghiệm



Không kinh nghiệm, các bạn sẽ cạnh tranh với nhau trong khoảng 7%.
Có kinh nghiệm, số lượng công việc sẽ rộng mở hơn nhiều

Vậy sinh viên kiếm kinh nghiệm ở đâu ra?

Cách đơn giản nhất là cố gắng đi thực tập vào năm 3. Việc đi làm sẽ dạy bạn được rất nhiều điều: làm dự án thực tế ra sao, quy trình thế nào, làm sao làm việc với khách hàng, đối xử với đồng nghiệp và cấp trên.

Tốt nhất là nên đi thực tập có lương, lương thấp cũng không sao, quan trọng là **môi trường tốt**, mình được làm nhiều, học hỏi được nhiều.

Nhà tuyển dụng cần kinh nghiệm để chứng tỏ bạn có khả năng, bạn làm được việc. Nếu không có cơ hội đi thực tập, bạn có thể làm freelance, làm các pet project hoặc dự án từ thiện.

Khi làm xong, bạn nhớ deploy dự án lên đâu đó, bỏ code lên GitHub; sau đó bỏ các dự án đó vào CV, đem khoe với nhà tuyển dụng để chứng tỏ năng lực nhé.

Tự học, tự trang bị kiến thức như thế nào?

Để không thất nghiệp, bạn cũng phải biết cách tự học để trang bị kiến thức cho bản thân mình.

Kiến thức trong ngành mình rất rộng, tài liệu học cũng rất nhiều và đủ thứ thượng vàng hạ cám. Nếu không biết cách học, bạn sẽ **dễ bị loạn, mất thời gian** mà khó đạt được mục tiêu (Hồi mình tự học UI/UX, học Cloud hay tìm hiểu về Data Mining cũng thế).

Theo mình, để việc học trở nên hiệu quả, các bạn nên làm theo những bước sau đây:

1. **Xác định thứ cần học**: Đừng ôm đồm, hãy tập trung vào học một thứ trước. Đừng học Java, PHP, C# cùng một lúc mà hãy học một ngôn ngữ thôi! Chọn một ngôn ngữ, một framework, đào sâu vào, đến khi nào thành thạo thì bắt đầu học cái mới.
2. **Xác định con đường**: Ví dụ muốn làm front-end dev, bạn cần tìm hiểu về HTML/CSS/JS cơ bản, rồi mới bắt đầu tìm hiểu các framework. Làm gì cũng vậy, nên học kiến thức nền trước rồi nâng cao dần, chứ đừng đâm đầu vào học framework ngay.
3. **Xem lại bản thân**: Nhớ xem lại những gì mình đã biết, bổ sung những gì mình chưa biết. Đừng nghĩ rằng mình đã biết hết! Đến bản thân mình, lâu lâu những thứ mình **tưởng rằng đã biết rồi** vẫn thấy thiếu hụt tùm lum, phải học lại để bổ sung này!

Nói chung, dù bạn có học cách nào thì **cách tốt nhất vẫn là làm**. Bạn có xem 3-4 khoá học, đọc hết 3-4 cuốn sách mà không viết dòng code nào thì cũng **không tiếp thu được**. Nhớ vừa học, vừa code, vừa áp dụng để tiếp thu kiến thức nhé!

Kết

Hi vọng bài viết này đã giúp bạn đỡ "bối rối" hơn, biết được phần nào những việc mình cần làm để lúc ra trường có thể tìm được một công việc ưng ý. Bạn nào có kinh nghiệm muốn chia sẻ thì cứ đăng trong phần comment nhé!

Bonus: Các bạn có thể tìm hiểu thêm qua slide thuyết trình và video phần trình bày của mình về chủ đề "Sinh viên IT học và làm gì để không thất nghiệp" nhé. Nếu chưa follow fanpage thì nhớ like và follow để đọc những bài viết hay mỗi ngày nghen.



30s quảng cáo



Những điều đề cập trong bài viết (scrum, học tiếng Anh) đều được giải thích rõ trong cuốn sách "Code dạo kí sự – Lập trình viên đâu phải chỉ biết code" do mình viết.

Quyển sách bao gồm những kĩ năng từ mềm đến cứng mà mỗi developer phải có, đảm bảo sẽ rất có ích cho các bạn sinh viên hoặc lập trình viên đã đi làm. Các bạn xem thông tin và đặt mua sách tại đây nhé: Sách Code Dạo Ký Sự.

Làm sao để trở thành một lập trình viên "có giá, lương cao"?

Là một lập trình viên, hẳn bạn nào cũng muốn có một công việc với mức lương khá khẩm, môi trường làm việc ngon lành.

Tuy nhiên, các công ty trả lương cho bạn dựa theo giá trị của bản thân bạn, tức là việc **bạn có thể mang lại bao nhiêu tiền cho công ty**. Muốn có mức lương như ý, bạn phải là một lập trình viên "có giá", đem lại nhiều giá trị cho công ty và cho team.

Vậy, phải làm sao để **nâng cao giá trị bản thân**, trở thành một lập trình viên "có giá"? Hãy đọc và làm theo những kinh nghiệm mình chia sẻ trong bài viết này nhé!

Trau đồi kĩ năng cứng

Công việc của lập trình viên không chỉ có code! Tuy vậy, thời gian code chiếm phần lớn thời gian làm việc của bạn.

Trau dồi kinh nghiệm và kĩ năng cứng là một trong những cách nhanh nhất để nâng cao giá trị bản thân.

Những việc bạn cần làm để trau dồi kĩ năng cứng là:

- **Nâng cao chất lượng code**: Hãy đọc Code Complete và Clean Code. Ngoài ra, hãy tìm hiểu thêm về các khái niệm chuyên sâu như: Nguyên lý SOLID, Dependency Injection, Design pattern. Hãy nâng tầm suy nghĩ lên tầm design, tầm hệ thống.
- **Học tiếng Anh**: Tiếng Anh tốt sẽ giúp bạn dễ đọc tài liệu, học công nghệ mới. Ngoài ra, bạn còn có cơ hội đi nước ngoài on-site, hoặc làm việc công ty nước ngoài, nhận mức lương cao. (Mình cũng từng chia sẻ một số kinh nghiệm học tiếng Anh nhé).
- **Tự bổ sung kiến thức**: Kiến thức trong ngành lập trình **rất nhanh hết hạn**. Đừng chỉ làm việc mình được giao, coi chừng kiến thức của bạn sẽ lạc hậu khi đi ra ngoài phỏng vấn xin việc đấy!

- Hãy bớt vào vozforum, webtretho, facebook lại mà chịu khó tìm đọc các blog IT, Pluralsight, Quora, Medium để tìm bổ sung kiến thức.



Kiến thức trong ngành lập trình rất rộng và rất dễ lỗi thời

- **Học Domain Knowledge**: Domain Knowledge tức là những kiến thức liên quan đến business, đến chuyên ngành (kinh tế, tài chính).
- Biết domain knowledge, các bạn có thể **hiểu điều khách hàng nói**, biết cách nói cho họ hiểu. Điều này tạo nên sự khác biệt, làm bạn "có giá" hơn. Bạn cũng có thể phát triển lên tầm BA – Business Analyst.
- **Tìm hiểu rõ dự án**: Có kĩ năng technical là tốt! Nhưng phải hiểu dự án mới biết cách áp dụng kĩ năng đem lại hiệu quả cao nhất!
- Hãy tìm hiểu kĩ về công nghệ, về scope và deadline, về những người chịu trách nhiệm chính trong dự án để có thể **đưa ra những đóng góp hữu ích** cho team.

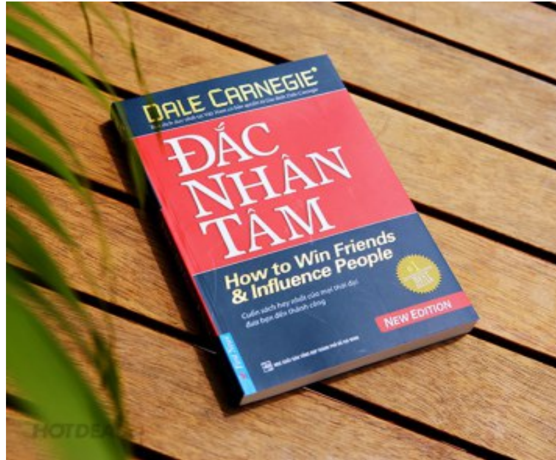
Phát triển kĩ năng mềm

Không chỉ các bạn sinh viên ít lưu tâm tới kĩ năng mềm, mà nhiều lập trình viên đã đi làm cũng coi thường tầm quan trọng của nó.

Nếu kĩ năng cứng, kĩ năng xin việc quyết định chuyện bạn có việc làm hay không; kĩ năng mềm sẽ quyết định **khả năng sống sót** với nghề, **khả năng thăng tiến** của bạn.

Những kĩ năng mềm mà các bạn nên phát triển là:

- **Kĩ năng giao tiếp**: Dân IT thường làm việc với máy nên không để ý đến kĩ năng giao tiếp. Thật ra, lập trình là làm việc với con người: Bạn sẽ phải trình bày với đồng đội, với sếp, với khách hàng.
- Dù cho bạn định theo hướng quản lý, khi lên vị trí cao như senior, software architect, bạn vẫn cần những **kĩ năng thuyết trình, kĩ năng diễn đạt.**



Mình khuyến khích các bạn nên đọc Đắc Nhân Tâm – Một cuốn sách khá hay về đối nhân xử thế và giao tiếp

- **Thương sếp và hiểu sếp**: Tại sao phải thương sếp hiểu sếp? Bởi vì việc bạn lên chức hay lên lương phần lớn là **do sếp quyết định**.
- Hãy thương sếp vì ngày xưa sếp cũng từng là dev như bạn, giờ làm quản lý họ cũng phải học hỏi từ đầu. Hãy hiểu sếp vì họ còn có nhiều chuyện phải lo hơn (dự án, tiến độ, lợi nhuận), không chỉ tập trung vào technical được nữa.
- **Quản lý thời gian**: Mỗi người chỉ có 8 tiếng mỗi ngày để làm việc. Nếu không biết cách quản lý, bạn sẽ khó hoàn thành công việc, dẫn đến OT. Mình có chia sẻ một bài viết về cách quản lý thời gian bằng Trello, các bạn tìm đọc nhé.
- Xây dựng uy tín và quan hệ: Ai cũng biết uy tín và quan hệ rất quan trọng.
- Uy tín giúp bạn đạt được nể trọng của sếp và đồng nghiệp, dễ lên lương lên chức. Quan hệ giúp bạn có nhiều cơ hội mới. Nhiều bạn chỉ cắm mặt lo làm mà quên mất hoặc không biết cách xây dựng hai thứ này.
- Bạn có thể dần dần xây dựng uy tín thông qua những việc nhỏ như: Tôn trọng deadline, hứa là làm; code có tâm, ít bug; không ngại việc "hơi" quá khả năng; sẵn sàng giúp đỡ đồng đội và junior.



Uy tín và thương hiệu cá nhân đóng vai trò khá quan trọng trên con đường thăng tiến của bạn

Về chuyện tăng lương

Một vấn đề mà developer chúng ta thường hay lăn tăn đó là chuyện "tăng lương". Các bạn nên hiểu rằng, công ty trả lương cho bạn theo khả năng, theo công việc bạn hoàn thành.

Nếu bạn chỉ hoàn thành những công việc được giao, khả năng của bạn không tăng lên, công ty **không có lý do gì để tăng lương cho bạn cả**!

Để dễ dàng "đòi hỏi" tăng lương, các bạn hãy đưa ra những lý do chính đáng, phù hợp như sau:

- Nói rõ những **cống hiến của bản thân cho công ty** và cho thành công của dự án
- Nâng cao khả năng, trình độ của mình bằng cách trau dồi các kĩ năng mềm cứng.
- Nhận nhiều trọng trách hơn, nhận trách nhiệm nhiều hơn thì dĩ nhiên **lương cũng sẽ tăng lên theo**
- Thử đi phỏng vấn bên ngoài, sau đó lấy offer về thương lượng.

Một kinh nghiệm khác khi thương lượng lương bổng là **đừng tin lời hứa của các sếp**. Hãy đòi hỏi họ **viết rõ ràng** về điều kiện tăng lương, tăng chức, **gửi qua email** để có cái làm bằng chứng sau này.



Đừng quá tin tưởng lời hứa suông của sếp, hãy đòi giấy tờ email xác thực

Ngay cả ở các công ty lớn như Amazon, FPT,... chuyện hứa lèo để giữ chân nhân viên, quịt rồi không tăng lương cũng không phải là hiếm nhé!

Nhảy việc nhiều lương sẽ cao?

Có khá nhiều lý do để các bạn nhảy việc: Công việc nhàm chán, lương thấp, không có cơ hội phát triển…

Nhảy việc có một số cái lợi: Khi nhảy việc, lương sẽ **tăng nhanh hơn** so với việc… chờ tăng lương. Điều này giúp bạn không bị hớ lương. Kể cả khi bạn không định nhảy việc, có offer mức lương cao hơn từ công ty khác thì bạn cũng **dễ nói chuyện với sếp hơn** để đòi tăng lương.

Tuy nhiên, nhảy việc có một số tác hại các bạn nên chú ý: Nó gây thiệt hại cho công ty (mất phí đào tạo bỏ ra cho bạn). Nhảy việc quá thường xuyên sẽ **làm xấu CV của bạn** (nhân viên nhảy nhiều thì công ty ít dám tuyển), đồng thời cũng **làm giảm cơ hội thăng tiến** của bạn.



Chuyện nhảy việc có những lợi ích và tác hại của nó

Do đó, lời khuyên của mình là nên **stay khoảng 2-3 năm** trước khi nhảy. Nên nhảy nếu tăng hơn 2-30% (Lương 10 lên 13tr chẳng hạn), tăng chỉ có mấy trăm nghìn thì đừng nhảy mất công.

Nếu nhắm thấy có thể phát triển lâu dài trong công ty hiện tại thì bạn có thể bám trụ lâu năm, đổi mức lương thấp lấy cơ hội thăng tiến về sau.

Ngoài ra, đừng nên nhảy việc khi dự án **đang thiếu nhân sự** hoặc đang **bước vào giai đoạn quan trọng** nhé. Bạn sẽ đẩy cấp trên và đồng đội vào thế khó xử, sau này rất khó nhìn mặt nhau!

Ngành IT mình nhỏ lắm, những chuyện như vậy sẽ ảnh hưởng lớn đến uy tín và hình ảnh của bạn sau này. Nhớ cẩn trọng nhé!

Kết

Bài viết này được tổng hợp từ nội dung buổi thuyết trình của mình tại FPT Software Hồ Chí Minh. Các bạn có thể tham khảo nội dung slide lại đây nhé: xem Slide.



Lời kết

Cuốn sách Nhập Môn Lập Trình Không Code cuối cùng cũng đi đến hồi kết.

Do bản chất nó chỉ là Nhập Môn, mình cố gắng không viết quá sâu về kĩ thuật. Vì vậy, series rất dễ nên **các em cấp 3, các bạn sinh viên năm nhất hoặc không học ngành lập trình** cũng có thể hiểu được.

Thông qua ebook này, mình đã chia sẻ về công việc của lập trình viên, những tố chất cần có, con đường trở thành một lập trình viên thực thụ. Hi vọng cuốn sách đã giúp các bạn có cái nhìn rõ ràng hơn về ngành lập trình, giúp bạn **định hướng được bản thân, biết con đường mình phải đi** trong tương lai.

Một người thầy giỏi không phải là người thầy **dạy cho bạn mọi thứ**, mà là người thầy dạy cho bạn **cách tự tìm hiểu mọi thứ**. Đọc xong cuốn sách này, tuy không viết được một dòng code nào, nhưng bạn sẽ biết **những gì mình cần phải tự học và tìm hiểu**, để dựa vào đó mà tự trau dồi kiến thức chính mình.

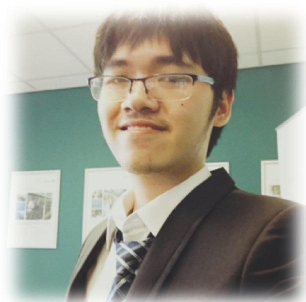
Cảm ơn các bạn đã bỏ thời gian đọc hết cuốn sách, chúc các bạn may mắn trên con đường IT mình đã chọn. Nếu có thắc mắc trong quá trình đọc, các bạn có thể thoải mái liên hệ mình hoặc fanpage Tôi đi code dạo để hỏi. Do mình lười nên nếu bạn không đọc bài, hỏi lại những câu hỏi mình đã trả lời rõ trong bài viết thì mình sẽ **bắt bạn đọc lại sách** đấy nhé ;).

P/S: Đây là ebook miễn phí, các bạn cứ thoải mái chia sẻ cho bạn bè, người thân, nhớ dẫn nguồn toidicodedao.com là được nhé. Để ủng hộ tác giả, nhớ ghé thăm và like fanpage tại: https://www.facebook.com/toidicodedao nhe.

Các bạn hãy click **Đăng kí nhận email** để theo dõi blog và nhận những **ebook miễn phí**, những **bài viết cực kì hay ho hàng tuần** về kĩ năng mềm và cứng, kinh nghiệm trong ngành lập trình nhé!

Đăng kí nhận email

Về tác giả



Mình là Phạm Huy Hoàng, hiện đang theo học tại Thạc sĩ về Khoa Học Máy Tính (Computer Science) tại Đại học Lancaster, Anh. Hiện mình đang làm ở vị trí Full-stack Developer cho phòng IT của trường.

Mình cũng là chủ blog Tôi Đi Code Dạo. Blog của mình chia sẻ về những kĩ năng cứng, kĩ năng mềm mà các lập trình viên cần phải có. Các bạn nhớ theo dõi vào thứ 3 và thứ 5 mỗi ngày nhé!

Thông tin liên lạc:

Email: huyhoang8a5@gmail.com
Blog: https://toidicodedao.com
Linkedin: https://www.linkedin.com/in/huyhoangpham92
CV: http://cv.toidicodedao.com

Sách đã ra mắt



Code dạo kí sự – Lập trình viên đâu phải chỉ biết code

Những phương cách trau dồi kĩ năng mềm, kĩ năng cứng đều được mình đề cập tới trong cuốn sách

Đây là quyển sách đầu tiên đề cập những kĩ năng từ mềm đến cứng mà mỗi developer phải có, đảm bảo sẽ rất có ích cho các bạn sinh viên hoặc lập trình viên đã đi làm.

Các bạn xem thông tin và đặt mua sách tại đây nhé: Sách Code Dạo Ký Sự.



Bảo Mật Nhập Môn – Bảo mật cơ bản cho developer

Những kiến thức vô cùng cơ bản về bảo mật mà developer nào cũng cần phải biết để bảo vệ website.

Với giọng văn hài hước, hóm hỉnh, dễ đọc dễ hiểu, cùng nhiều ví dụ minh hoạ, ebook đã được nhiều bạn đọc đón nhận, với hơn 15 ngàn lượt tải.

Bạn vào http://security.toidicodedao.com để tải ebook miễn phí nhé!